Journal of ELECTRICAL ENGINEERING, VOL. 61, NO. 2, 2010, 120–124

GENERALIZATION OF PATTERNS BY IDENTIFICATION WITH POLYNOMIAL NEURAL NETWORK

Ladislav Zjavka *

Artificial neural networks (ANN) in general classify patterns according to their relationship, they are responding to related patterns with a similar output. Polynomial neural networks (PNN) are capable of organizing themselves in response to some features (relations) of the data. Polynomial neural network for dependence of variables identification (D-PNN) describes a functional dependence of input variables (not entire patterns). It approximates a hyper-surface of this function with multi-parametric particular polynomials forming its functional output as a generalization of input patterns. This new type of neural network is based on GMDH polynomial neural network and was designed by author. D-PNN operates in a way closer to the brain learning as the ANN does. The ANN is in principle a simplified form of the PNN, where the combinations of input variables are missing.

K e y w o r d s: polynomial neural network, dependence of variables identification, rational fractional functions, function approximation, differential equation, modelling of complex systems

1 INTRODUCTION

Artificial neural network (ANN) is trained to classify certain patterns into groups, and then is used to classify the new ones, which were never presented before. ANN can correctly identify incomplete or similar patterns as compared to the training data set [5]. N-dimensional vector of variables, which defines a pattern or marks a state of system, is multiplied by weights of each neuron the ANN, Fig. 1. The relative dependence of the input variables (if there is any) will be lost subsequently.

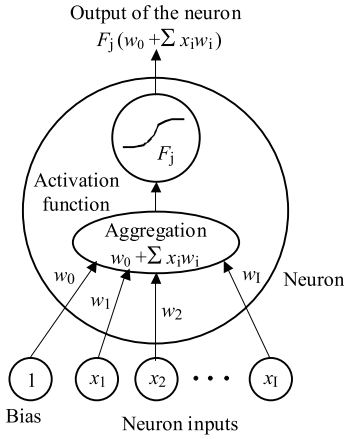

Fig. 1. Artificial neuron

Let us try to look at the vector of input variables as a no "pattern but dependent bounded characteristic point set. Relations to each other can describe a very complicated multi-parametric function. The idea here, is to create a neural network, which can identify (find and learn) any unknown dependencies of the input vector variables. The neural network response should be the same to all patterns (sets), which behave the trained dependence. Their values can differ enormously, so it is necessary to treat with the relative ones. This could be regarded as a pattern abstraction, where classification is not based on values of variables (forming a pattern) but only relations of these. Polynomial neural network for dependence of variables identification (D-PNN) forms the functional output as a generalization of input patterns.

2 GMDH POLYNOMIAL NEURAL NETWORK

Starting point of this new type of neural network development was the GMDH polynomial neural network, Fig. 2. In the hope to capture the complexity of a process, this neural network attempts to decompose it into many simpler relationships each described by a processing function of a single neuron. It was created by a Ukrainian scientist Aleksey Ivakhnenko in 1968, when back-propagation technique was not known yet [5].

$$y = a_0 + \sum_{i=1}^{m} a_i x_i + \sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij} x_i x_j + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} a_{ijk} x_i x_j x_k + \dots \quad (1)$$

m – number of variables,
$X(x_1, x_2, \dots, x_m)$ – vector of input variables,
$A(a_1, a_2, \dots, a_m)$, … – vectors of parameters.

He attempted to resemble the Kolmogorov-Gabor polynomial (1), which defines general connection between input and output variables, by using low order polynomials (2) for every pair of input values [2]

$$y = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j + a_4 x_i^2 + a_5 x_j^2. \quad (2)$$

A technique called Group Method of Data Handling (GMDH) was developed for neural network structure design and parameters of polynomials adjustment.

* Department of Informatics, Faculty of Management Science and Informatics, University of Žilina, Univerzitná 8215/1, 010 01 Žilina, Slovakia; lzjavka@gmail.com

ISSN 1335-3632

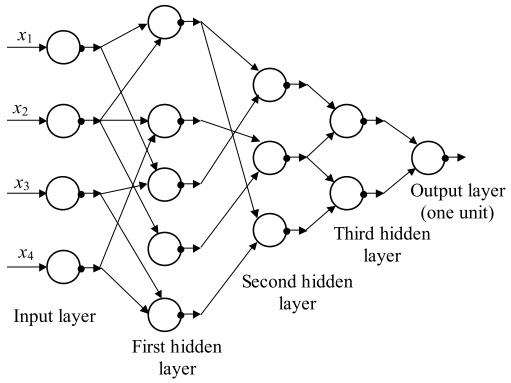

Fig. 2. GMDH polynomial neural network

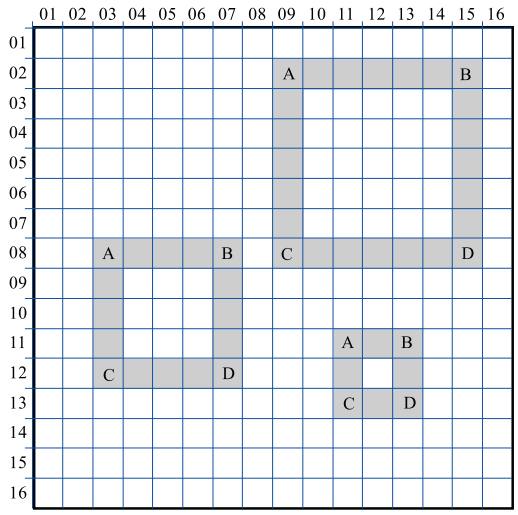

Fig. 3. Characteristic points of an alternate pattern

The GMDH neuron has two inputs and its output is a quadratic combination of 2 inputs (total 6 weights). Thus GMDH network builds up a polynomial (actually a *multinomial*) combination of the input components [3]. Typical GMDH network maps a vector input $\boldsymbol{x}$ to a scalar output y, which is an estimate of the true function $f(\boldsymbol{x}) = y$. Each neuron of the polynomial network fits its output to the desired value y for each input vector $\boldsymbol{x}$ from the training set. The manner in which this approximation is accomplished is through the use of linear regression [6].

The network structure is evolved during estimation inductive self-organizing process in comparison with the ANN, which structure must be given in advance. GMDH polynomial neural network creates a structural model of complex system. It is used especially for time-series prediction of systems with dependent variables, which are described only with small input-output data samples [1].

3 GENERALIZATION OF PATTERNS PROBLEM FORMULATION

ANN can correctly classify any new input patterns according to their relationship. But if a geometric shape (forming an input pattern) is enlarged, made smaller or moved, it will seem to the neural network to be an entirely new pattern. There are total *different areas of neurons* activated, in comparison with the training data set classification [9]. One way how we can resolve this problem is to define several characteristic points of the pattern (forming the input vector of the D-PNN) and try to obtain the dependence of these. This dependence defines multiparametric function, creating by particular polynomials of the D-PNN, applied to general pattern identification.

You can see, that the condition of the significant point pattern dependence (Fig. 3.) is defined by equal x-coordinates of the points A, B and C, D and equal y-coordinates A, C and B, D as a rectangle. Another more complicated diagonal dependence applies the coordinates of points A, C and B, D to oblique square determination. There will have to equal sum and difference their x and y-positions. Together in this 2-way manner dependence a square shape is distinguished D-PNN. According to latest human brain researches, recognizing shapes are decomposing into several elementary elements, which are activating some biological neurons as characteristic marks of the pattern. Human brain does not treat with absolute input values but relative reciprocal ones, which are creating by periodic dynamic functions.

4 DEPENDENCE OF VARIABLES IDENTIFICATION

The basic idea of authors D-PNN is to approximate a differential equation (3), which can define the relations of variables, with a special type of root fractional polynomials — for instance (4). After the structure and parameter adjustment, the output of the D-PNN should be the same to all input vectors, which variables behave the unknown learned dependence of the training data set (it does not matter on their values).

$$Y = a + \sum_{i=1}^{n}\sum_{j=1}^{n-1} b_{ij}\frac{\partial x_i}{\partial x_j} + \sum_{i=1}^{n}\sum_{j=1}^{n-1}\sum_{k=1}^{n-1} c_{ijk}\frac{\partial^2 x_i}{\partial x_j \partial x_k} + \dots \quad (3)$$

a, $B(b_{11}, b_{12}, \dots, b_{nn-1})$, ... - - parameters.

Elementary methods of common differential equation solution express the solution with special elementary functions — polynomials (such as Bessels functions or power series). Numerical integration of differential equations is based on approximation of these with [4]:

- rational integral functions,
- trigonometric series.

Here a more simple way using method of integral analogues was selected, the 1st , by replacing mathematical operators in equations with ratio of pertinent values [7].

$$y = \frac{1}{(b_0 + b_1x_1 + b_2x_2 + b_3x_1x_2)^{1/2}}\big(a_0 + a_1x_1 + a_2x_2 + \dots a_{n+1}x_1x_2 + \dots + a_{n+m}x_1x_2x_3 + \dots\big)^{1/(n-1)} \quad (4)$$

n – degree of combinations.

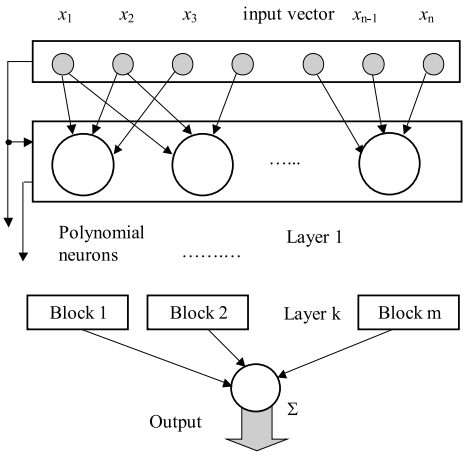

Fig. 4. Polynomial neural network for dependence of variables identification D-PNN

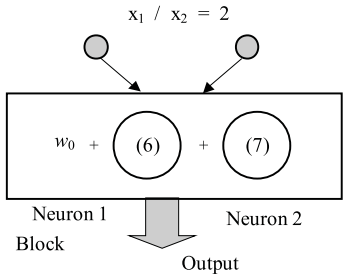

Fig. 5. Identification of a constant quotient of 2 variables

The root fractional polynomials (4), which describe the dependence of input variables, are applied as terms of differential equation (3). These polynomials are partly creating an unknown multi-parametric non-linear function and replace it. The formula in numerator of equation (4) is a polynomial of complete n-degree combination of inputs and substitutes the input variables into a new transformation function z. The denominator of equation (4) is a derivative term, which gives the particular mutual change of some variables and its polynomial combination degree is less then n. Its root function is lower too (according to the combination degree). In general it is possible this approximation express in formula (5):

$$Y = w_0 + w_1\frac{\partial z}{\partial x_1} + w_2\frac{\partial z}{\partial x_2} + \dots + w_n\frac{\partial z}{\partial x_n} + w_{n+1}\frac{\partial^2 z}{\partial x_1 \partial x_2} + \dots \quad (5)$$

z – function of n-input variables,
w_i – weights of terms.

This function (5) with complete root polynomials, replacing the transformation function z of numerator, is formed in the last hidden layer of the D-PNN. The parts of fractions (4) interfere (compensate) each other. There are only ordinary polynomials in the other hidden layers, without derivative terms (in dominator). The inputs from previous layers can enter next layers, so can influence to combination creation of which. These inputs can be applied as derivative terms by creating of differential equation (5) in the last hidden layer too. Another way we can create a particular differential equation of the each layer as a part of the D-PNN total output. This double-layer contains also ordinary polynomial neurons, which output enters the next layer. The fractional polynomials of function approximation take part only in the total output. Each next layer should eliminate the previous layer output error and approach D-PNN to the best result.

The D-PNN (Fig. 4) consists of polynomial neurons and blocks of neurons, where each block represents a single differential equation (5). Neurons are creating particular terms of the equations applying fractional polynomials (4). Inputs of constant combination degree $(2, 3, \dots)$ enter each block or polynomial neuron. Each neuron has 2 vectors of adjustable parameters $\boldsymbol{a}$, $\boldsymbol{b}$ and a single weight as a term of differential equation. The root functions (4) linearize the multiplications (combinations) of dependent variables in polynomials, creating an unknown exponential function. D-PNN has to approximate these unknown quadratic functions, which are hidden in the dependent training data set and define the dependence of its variables.

It is necessary to adjust not only polynomial parameters, but the D-PNN structure too. The GMDH polynomial neural network is adjusted as well like this way, by trimming of extraneous (big error output) neurons and changed the structure of polynomials to fit it to the best total output [5].

5 EXPERIMENTAL PART

Consider a very simple dependence of 2-input variables, which difference is constant (for example 5). D-PNN will learn this relation according to training data set by means of genetic algorithm (GA).

$$y = w_1\frac{(a_0 + a_1x_1 + a_2x_2 + a_3x_1x_2)^{1/2}}{b_0 + b_1x_1}. \quad (6)$$

D-PNN will consist only 1 block of 1 polynomial neuron (6) as a term of differential equation (like Fig. 5). As the input variables are changing constantly, there is not necessary to add the 2^{nd} term (fractional polynomial of derivative variable x2) in the differential equation (block) and causes occasional output mistakes by identification (GA have finished in a local error minimum). Another example can solve 2 inputs, where the multiplicity of the inputs is constant (Fig. 5).

The input variables are not changing constantly, so there will be necessary both terms of the differential equation - fractional polynomial of derivative variable x_2 too (7)

$$y = w_2\frac{(a_0 + a_1x_1 + a_2x_2 + a_3x_1x_2)^{1/2}}{b_0 + b_1x_2}. \quad (7)$$

Simple D-PNN composed of these 4 blocks (one block for each pair of input variables) can learn the characteristic point dependence of the rectangle shape (see Fig. 3). If

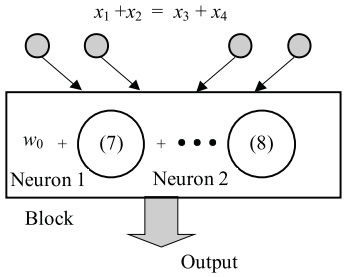

Fig. 6. Identification of a 4-variable dependence

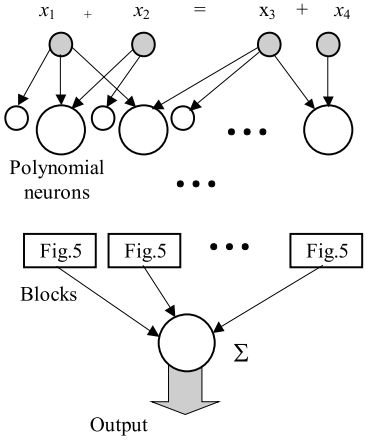

Fig. 7. Identification of the 4-variable dependence with 2-degree combinations

x-coordinates of the points A, B and C, D equal and y-coordinates A, C and B, D equal, their difference = 0 and D-PNN will learn and identify this. Another dependence of the oblique square shape uses the diagonal coordinates of points A, C and B, D. If the x and y-position difference of the 1st couple of points (4 variables) equals and the x and y-position sum of the 2nd ones equals, the oblique square point dependence condition comes true. This can resolve 2 blocks of 4 input variables D-PNN (Fig. 6).

There are total 14 combinations (neurons) of 4 input variables, for all derivative terms (combination degrees 1,2,3) in the block. Some of them have to be taken off, as cause the D-PNN will work amiss. A neuron can form for example the fraction (8)

$$y = w_i \frac{1}{(b_0 + b_1 x_1 x_2 + b_2 x_2 + b_3 x_1 x_2)^{1/2}} \big(a_0 + a_1 x + \cdots + a_5 x_1 x_2 + \cdots + a_{11} x_1 x_2 x_3 + \cdots + a_{15} x_1 x_2 x_3 x_4\big)^{1/4} . \quad (8)$$

The D-PNN as well is charged by possible 2-sided changes of input variables. For example $1 + 9 = 10$ is the same sum as $9 + 1 = 10$. Additional blocks of neurons could help with this problem. There will occur some mistakes by identification, if GA adjustment finishes in a local error minimum.

This problem could resolve D-PNN with 2-degree combinations of variables (Fig. 7.), even this way more mistakes occur than the previous one. But it can be clearly seen that the D-PNN operates.

Table 1. 2-leveled composite functions of the sum dependence

x_1	+	x_2	=	x_3	x_1x_2	x_1x_3	x_2x_3	$x_1x_2x_3$
1		1		2	1	2	2	2
1		2		3	2	6	3	6
1		3		4	3	12	4	12
2		2		4	4	8	8	16
1		4		5	4	20	5	20
2		3		5	6	15	10	30
1		5		6	5	30	6	30
2		4		6	8	24	12	48
3		3		6	9	18	18	54
1		6		7	6	42	7	42
2		5		7	10	35	14	70
3		4		7	12	28	21	84
1		7		8	7	56	8	56
2		6		8	12	48	16	96
3		5		8	15	40	24	120
4		4		8	16	32	32	128
1		8		9	8	72	9	72
2		7		9	14	63	18	126
3		6		9	18	54	27	162
4		5		9	20	45	36	180

The compound function which describes the dependence of variables (Tab. 1) is 2-leveled (or more). Main exponential function carries some secondary functions. Therefore the solution of this D-PNN will comprise additional hidden layers of polynomial neurons, which represent the main (outer) carrying functions. Then the 1st hidden layer replaces the secondary functions of the dependence. From mathematical point of view the 1st layer forms the inside functions and the 2nd layer utilizes these to form the outside power functions. D-PNN will contain more than 3 hidden layers in this case.

The inputs from previous layers can enter each next layer, so they can be applied to combination creation of another polynomials and blocks. Number of combinations is increasing in this case. Thoroughly it is necessary to adjust the structure of the D-PNN, only some neurons of some possible combinations will be utilized. This way we can recover again the D-PNN of higher combination degree each next hidden layer (likewise the previous example).

6 DISCUSSION

D-PNN is new authors neural network type, which operates in a more similar way to the brain learning as the ANN (or PNN) does. Human brain does not treat with absolute input values but the relative reciprocal ones, where the transformation is done by periodic dynamic functions. D-PNN identification is not based on pattern similarity, but only the same unknown dependencies of the input vector variables. If there is any dependence of variables in a training data set, D-PNN should found and learn it. The values of input vector variables can differ largely therefore it is necessary to eliminate them by fractional functions. Instead of these some periodic

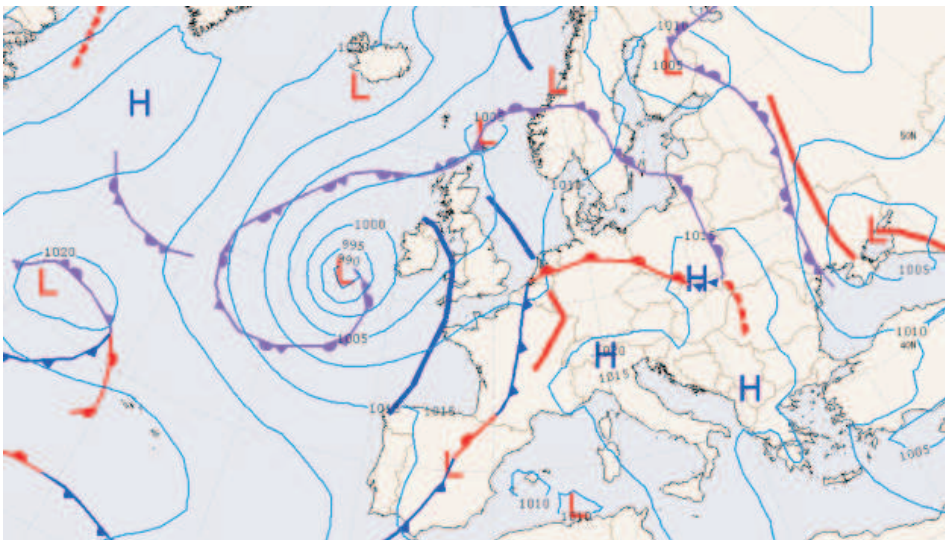

Fig. 8. Meteorological pressure map

functions could be used (sin, cos) for this transformation. Changeable periods ω will replace the derivative terms of equations — for instance (4) in denominator in this case [7].

The method of input pattern generalization creates one abstract pattern described by polynomials for all changeable forms of shape (pattern) using some significant dependent points of the data vector sets. This could be regarded as a simulation of human brain learning, which is searching the input data too to find and generalize any dependencies of it. It applies but the approximation with periodic activation functions of biological neurons a dynamic system of behaviour [8]. D-PNN constructs a differential equation (which describes a system with dependent variables) of rational integral functions. The determination of characteristic points of a changeable pattern could cause some problems, but a special algorithm should solve these.

7 CONCLUSION

This way of identification could be applied as model of complex system with dependent variables, where some dependence of these leads into a new system state, for instance weather prediction. Pressure, damp and temperature maps (Fig. 8) with some significant points form input vector describing an actual state. Isobars of the same pressure (for instance) define patterns (shapes) and expressive curving of which could define some significant point sets.

Time-series prediction will not be applied in this case (it is based on pattern identification). Common GA as an emergency tool is applied for D-PNN structure design and parameter adjustment for the time being. In the future could be utilized learning schemes, which describe possible training ways of the D-PNN. These would be gradually created and improved on each next learning process instead of GA. A combine improved GA could be another way, partly utilizing the output error computing.

References

[1] IVACHNENKO, A. G.: Polynomial Theory of Complex Systems, IEEE Transactions on Systems, Man and Cybernetics **SMC-1** No. 4 (Oct 1971).

[2] VASECHKINA, E. F.—YARIN, V. D.: Evolving Polynomial Neural Network by Means of Genetic Algorithm, Complexity International **09** (July 2001).

[3] ANASTAKIS, L.—MORT, N.: The Development of Self-Organisation Techniques in Modeling: A Review of the GMDH, Research report num. 813, University of Sheffield, Oct 2003.

[4] NEUSCHL, Š.—BLATNÝ, J.—ŠAFAŘÍK, J.—ZENDULKA, J.: Modellng and Simulation (Modelovanie a simulácia), ALFA, Bratislava, 1988. (in Slovak)

[5] GALKIN, I.: Polynomial Neural Networks. Materials for UML 91.531 Data Mining Course, University Mass Lowell. (http://ulcar.uml.edu/ iag/CS/Polynomial-NN.html).

[6] HOWLAND, J. C.—VOSS, M. S.: Natural Gas Prediction Using The Group Method of Data Handling, 7th International Conference on Artificial Intelligence and Soft Computing, Banff, Alberta 2003.

[7] KUNEŠ, J.—VAVROCH, O.—FRANTA, V.: Principles of Modelling (Základy modelování, SNTL, Praha, 1989. (in Czech)

[8] BEŇUŠKOVÁ, Ľ.: Neuron and Brain. Cognitive Sciences. (Neurón a mozog. Kognitívne vedy), Calligram, Bratislava, 2002. (in Slovak)

[9] ZJAVKA, L.: Polynomial Neural Network, Transcom 2007, 7-th European conference proceedings. Žilina June 25–27, 2007.

[10] ZJAVKA, L.: Dependence of Variables Identification with Polynomial Neural Network, Proceedings of CSE 2008 International Scientific Conference on Computer Science and Engineering. September 24–26, 2008 The High Tatras - Stará Lesná.

Received 19 June 2009

Ladislav ZJAVKA was born in 1966. After taking a degree from the University of Transport and Communication in Žilina, the Faculty of Engineering in 1989, he had worked as a programmer of database systems in several enterprises. Since year 2000 he has been employed as an assistant at the Faculty of Management Science and Informatics, University of Žilina. His scientific research is focusing on polynomial neural networks and genetic programming. In addition, he also investigates questions related to the human brain learning. Now he is an external PhD student.