# Application of Relative Derivation Terms by Polynomial Neural Networks

Ladislav Zjavka

VŠB – Technical University of Ostrava, IT4innovations Ostrava, Czech Republic ladislav.zjavka@vsb.cz

**Abstract.** A lot of problems involve unknown data relations, which can define a derivative based model of dependent variables generalization. Standard softcomputing methods (as artificial neural networks or fuzzy rules) apply usual absolute interval values of input variables. The new proposed differential polynomial neural network makes use of relative data, which can better describe the character regarding a wider range of input values. It constructs and resolves an unknown partial differential equation, using fractional polynomial sum derivative terms of relative data changes. This method might be applied to solve problems concerned a visual pattern generalization or complex system modeling.

#### **1** Introduction

Differential equations are able to solve a variety of pattern recognition and function approximation problems [2]. A principal lack of the artificial neural network (ANN) behavior in general is a disability of the data relation generalization [8]. Differential polynomial neural network (D-PNN) is a new neural network type, designed by the author, which creates and resolves an unknown partial differential equation (DE) of a multi-parametric function approximation. A DE is replaced producing sum of fractional polynomial derivative terms, forming a system model of dependent variables. Its regression is not based on a simple whole-pattern affinity but learned generalized data relations. This seems to be mainly profitable by application of different learning and testing interval values of input variables. Standard softcomputing methods usual are not able to operate correctly on varying training and testing data range, utilizing only the absolute values.

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

m – number of variables  $A(a_1, a_2, ..., a_m), ... - vectors of parameters <math>X(x_1, x_2, ..., x_m)$  - input vector

D-PNN resulted from the GMDH polynomial neural network (Fig.1.), which was created by a Ukrainian scientist Aleksey Ivakhnenko in 1968 [3]. When the back-

V. Snášel, K. Richta, J. Pokorný (Eds.): Dateso 2013, pp. 98-108, ISBN 978-80-248-2968-5.

propagation technique was not known yet a technique called Group Method of Data Handling (GMDH) was developed for neural network structure design and parameters of polynomials adjustment. General connection between input and output variables is expressed by the Volterra functional series, a discrete analogue of which is Kolmogorov-Gabor polynomial (1). This polynomial can approximate any stationary random sequence of observations and can be computed by either adaptive methods or system of Gaussian normal equations.

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

GMDH decomposes the complexity of a process into many simpler relationships each described by low order polynomials (2) for every pair of the input values. Typical GMDH network maps a vector input x to a scalar output y', which is an estimate of the true function f(x) = y. Each neuron of the polynomial network fits its output to the desired value y for each input vector x from the training set. It defines an optimal structure of complex system model with identifying non-linear relations between input and output variables [5].

#### 2 Differential polynomial neural network

The basic idea of the D-PNN is to create and replace a partial differential equation (DE) (3), which is not known in advance and is able to describe a system of dependent variables, with a sum of fractional multi-parametric polynomial derivative terms (4)[2].

$$a + \sum_{i=1}^{n} b_i \frac{\partial u}{\partial x_i} + \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} + \dots = 0 \qquad \qquad u = \sum_{k=1}^{\infty} u_k$$
(3)(4)

 $u(x_1, x_2, ..., x_n)$  - searched function of all input variables (dependent variable) a,  $B(b_1, b_2, ..., b_n)$ ,  $C(c_{11}, c_{12}, ...)$  - parameters

The applied method of integral analogues replaces math operators and symbols of a DE by ratio of corresponding variables. Derivatives are replaced by the integral analogues, i.e. derivative and all operators are replaced by analogous or proportion marks in equations [4].

$$u_{i} = \frac{\left(a_{0} + a_{1}x_{1} + a_{2}x_{2} + a_{3}x_{1}x_{2} + a_{4}x_{1}^{2} + a_{5}x_{2}^{2} + \ldots\right)^{m/n}}{b_{0} + b_{1}x_{1} + \ldots} = \frac{\partial^{m} f(x_{1}, \dots, x_{n})}{\partial x_{1}\partial x_{2} \dots \partial x_{m}}$$
(5)

n – combination degree of a complete polynomial of n-variables m – combination degree of denominator

The numerator of a term (5) is a polynomial of all *n*-input variables of a single neuron and partly defines an unknown function u of eq. (3)(4). The denominator is a derivative part of a DE term (5), which arose from the partial derivation of the complete *n*-variable polynomial by competent variable(s). The root function of numerator takes the polynomial into competent combination degree but needn't be used at all if not necessary.

A block of the D-PNN (Fig.1.) consists of basic neurons, one for each fractional polynomial (5), defining a sum partial derivative term of the DE (3) solution. Blocks of higher layers are additionally extended with compound neurons of composite functions, which apply previous layer block outputs and inputs. Each block contains a single output polynomial (without derivative part), thus the block skeleton of the D-PNN is formed by the GMDH network. Neurons don't affect the block output but are applied directly in the sum of a total output calculation of a PDE composition (4). Each block has 1 and neuron 2 vectors of adjustable parameters *a*, resp. *a*, *b*.



Fig. 1. D-PNN block of basic and compound neurons

Root mean square (RMS) error method (6) was applied for polynomial parameter optimization and PDE term selection.

$$E = \sqrt{\frac{\sum_{i=1}^{M} \left( y^d - y_i \right)^2}{M}} \to \min$$
(6)

#### 3 Multi-layered backward D-PNN

Multi-layered D-PNN forms composite polynomial functions (Fig.2.). Compound DE terms, i.e. derivatives in respect to variables of previous layers, are calculated according to the composite function partial derivation rules (7)(8). They are formed by products of partial derivatives of external and internal functions.

$$F(x_1, x_2, \dots, x_n) = f(y_1, y_2, \dots, y_m) = f(\phi_1(X), \phi_2(X), \dots, \phi_m(X)) \qquad i = 1, \dots, m$$
(7)

$$\frac{\partial F}{\partial x_k} = \sum_{i=1}^m \frac{\partial f(y_1, y_2, \dots, y_m)}{\partial y_i} \cdot \frac{\partial \phi_i(X)}{\partial x_k} \qquad k = 1, \dots, n$$
(8)

Each block of the D-PNN involves basic neurons e.g. (9), at first of only linear regression. Additionally blocks of the  $2^{nd}$  and following hidden layers are also extended with neurons, which form composite derivatives utilizing outputs and inputs

of back connected previous layer blocks, e.g. the  $1^{st}$  block of the last  $(3^{rd})$  hidden layer (10)(11) [7].

$$y_{1} = \frac{\partial f(x_{21}, x_{22})}{\partial x_{21}} = w_{1} \frac{\left(a_{0} + a_{1}x_{21} + a_{2}x_{22} + a_{3}x_{21}x_{22}\right)^{\frac{1}{2}}}{2 \cdot (b_{0} + b_{1}x_{21})}$$
(9)

$$y_{2} = \frac{\partial f(x_{21}, x_{22})}{\partial x_{11}} = w_{2} \frac{(a_{0} + a_{1}x_{21} + a_{2}x_{22} + a_{3}x_{21}x_{22})^{\frac{1}{2}}}{2 \cdot x_{22}} \cdot \frac{(x_{21})^{\frac{1}{2}}}{2 \cdot (b_{0} + b_{1}x_{11})}$$
(10)

$$y_{3} = \frac{\partial f(x_{21}, x_{22})}{\partial x_{1}} = w_{3} \frac{(a_{0} + a_{1}x_{21} + a_{2}x_{22} + a_{3}x_{21}x_{22})^{\frac{1}{2}}}{2 \cdot x_{22}} \cdot \frac{(x_{21})^{\frac{1}{2}}}{2 \cdot x_{12}} \cdot \frac{(x_{11})^{\frac{1}{2}}}{2 \cdot (b_{0} + b_{1}x_{1})}$$
(11)



Fig. 2. 3-variable 2-combination block D-PNN

The best-fit neuron selection is the initial phase of the DE composition and may apply a proper genetic algorithm (GA). Parameters of polynomials might be adjusted by means of difference evolution algorithm (EA), supplied with sufficient random mutations [1]. The parameter optimization is performed simultaneously with the GA term combination search, where may arise a quantity of local and global error solutions. There would be welcome to apply an adequate gradient descent method too, which parameter updates result from partial derivatives of polynomial DE terms in respect with the single parameters [6]. The number of network hidden layers coincides with a total amount of input variables.

$$Y = \frac{\sum_{i=1}^{k} y_i}{k} \qquad k = amount of active DE terms$$
(11)

Only some of all potential combination DE terms (neurons) may participate in the DE composition, in despite of they have an adjustable term weight ( $w_i$ ). D-PNN's total output Y is the sum of all active neuron outputs, divided by their amount k (11).

#### 4 Identification of data relations

Consider first only a linear simplification of data relations, thus only linear polynomials of neurons and blocks might be applied. D-PNN consisting of only 1 block of 2 neurons, all terms of the DE (12), is able to identify simple linear 2-variable dependence (function), e.g.  $x_1 = 2x_2$ .

$$y = w_1 \frac{\left(a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1 x_2\right)^{1/2}}{b_0 + b_1 x_1} + w_2 \frac{\left(a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1 x_2\right)^{1/2}}{b_0 + b_1 x_2}$$
(12)

More complicated dependence, where 2 variables depend on a  $3^{rd}$  (e.g.  $x_1 + x_2 = x_3$ ) may be resolved again D-PNN with one 3-variable combination block. The complete DE (of 1 and 2-combination derivatives) consists of 6 sum terms (neurons) but only 3 may be employed, derivative terms for  $x_3$  (13),  $x_1x_3$  (14),  $x_2x_3$  (15). Some neurons must be inactivated, having an undesirable effect on the network correct operation. The applied 2-variable combination block D-PNN has 3 hidden layers (Fig.2.).

$$y_1 = w_1 \frac{\left(a_0 + a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_1 x_2 + \dots + a_7 x_1 x_2 x_3\right)^{\frac{1}{3}}}{b_0 + b_1 x_3}$$
(13)

$$y_2 = w_2 \frac{\left(a_0 + a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_1 x_2 + \dots + a_7 x_1 x_2 x_3\right)^{2/3}}{b_0 + b_1 x_1 + b_2 x_3 + b_3 x_1 x_3}$$
(14)

$$y_{3} = w_{3} \frac{\left(a_{0} + a_{1}x_{1} + a_{2}x_{2} + a_{3}x_{3} + a_{4}x_{1}x_{2} + \dots + a_{7}x_{1}x_{2}x_{3}\right)^{\frac{2}{3}}}{b_{0} + b_{1}x_{2} + b_{2}x_{3} + b_{3}x_{2}x_{3}}$$
(15)

D-PNN can indicate the learned dependence of 3 variables (function) by the output value *1.0* (or any desired). It was trained with only 6 data samples (Tab.1), which were selected to involve proportionally the whole training data interval values <0,500>. However the output function values  $x_3=x_1+x_2$  (x-axis) of the test random input vectors can exceed the maximal trained sum value *500*, while the response is

kept (Fig.3.). Output errors can result from very disproportional random vector values, which D-PNN was not trained to, e. g. 360 = 358 + 2.

	1	2	3	4	5	6
<b>X</b> 1	1	70	40	160	30	300
<b>X</b> <sub>2</sub>	2	3	100	60	330	200
X3	3	73	140	220	360	500

**Table 1.** Training data set of the 3-variable dependence (function) identification  $x_1 + x_2 = x_3$ 

The identification of data relations might be applied to a generalization of fragmented visual patterns into some characteristic dependent elements, which shape assume moved or sized form in the input matrix and where ANN applications fail [7]. The outcomes of 1-block and multi-layered D-PNN are comparable, however the 2<sup>nd</sup> type is able to involve far larger amount of DE terms and so form a more accurately description of a model.



Fig. 3. Identification of a multi-parametric function relation

### 5 Function approximations

D-PNN can approximate a multi-parametric function, analogously to the ANN approach. Consider the sum function again  $y^t = x_1 + x_2 + x_3$ , however it could be any linear function. The network with 3 input variables, forming 1 output  $y = f(x_1, x_2, x_3)$  should approximate the true function  $y^t$  by means of sum derivative terms of the partial DE solution. The training data was necessary to be doubled into 12 samples (Tab.2). The D-PNN and ANN approximation is co-equal on the trained interval values <6, 520>, however the ANN approximation ability rapidly falls outside of this range (Fig.4.). The type and operating principle of the D-PNN is the same with

applied the dependence identification (Fig.2.), though requiring more time-consuming adjustment.

	1	2	3	4	5	6
<b>X</b> 1	1	70	4	160	200	30
<b>X</b> <sub>2</sub>	2	3	100	90	20	330
<b>X</b> <sub>3</sub>	3	20	40	10	100	20
$\mathbf{y}^{\mathbf{d}}$	6	93	144	260	320	380
	7	8	9	10	11	12
x <sub>1</sub>	4	10	150	20	50	260
<b>X</b> <sub>2</sub>	5	70	5	210	150	60
X3	12	80	55	100	200	200
yd	21	160	210	330	400	520

**Table 2.** The  $y^t = x_1 + x_2 + x_3$  function approximation training data set



Fig. 4. Comparison of a linear multi-parametric function approximation

$$F\left(x, y, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial^2 u}{\partial x^2}, \frac{\partial^2 u}{\partial x \partial y}, \frac{\partial^2 u}{\partial y^2}\right) = 0$$
(16)

where F(x, y, u, p, q, r, s, t) is a function of 8 variables

In the case of a real-data application D-PNN processes 2-combination square polynomials of blocks and neurons (DE terms), the same as applied by the GMDH algorithm (2). This simple polynomial type proves to yield best results besides an easy use and improves also the linear function approximation (which is notable). Thus each block includes 5 basic neurons of derivatives  $x_1$ ,  $x_2$ ,  $x_1x_2$ ,  $x_1^2$ ,  $x_2^2$  of the 2<sup>nd</sup>

order partial DE (3) of an unknown 2-variable function u, which might be transferred into form of eq. (16). Without this extension only a linear regression of the training data set would be applied. The square and combination derivative terms are also calculated according to the composite function derivation rules (17)(18). However they don't apply the complete sum of the formulas but only 2 simple terms with 1<sup>st</sup> order external function derivatives e.g. (19).

$$F(x, y) = f(u, v) = f[\varphi(x, y), \psi(x, y)]$$
(17)

$$\frac{\partial^2 F}{\partial x^2} = \frac{\partial^2 f}{\partial u^2} \left( \frac{\partial \varphi}{\partial x} \right)^2 + 2 \frac{\partial^2 f}{\partial u \partial v} \frac{\partial \varphi}{\partial x} \cdot \frac{\partial \psi}{\partial x} + \frac{\partial^2 f}{\partial v^2} \left( \frac{\partial \psi}{\partial x} \right)^2 + \frac{\partial f}{\partial u} \cdot \frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial f}{\partial v} \cdot \frac{\partial^2 \psi}{\partial x^2}$$
(18)

$$y_{4} = \frac{\partial^{2} f(x_{21}, x_{22})}{\partial t_{11}^{2}} = w_{4} \frac{(a_{0} + a_{1}x_{21} + a_{2}x_{22} + a_{3}x_{2}x_{22} + a_{4}x_{21}^{2} + a_{5}x_{22}^{2})^{\frac{1}{2}}}{3 \cdot x_{22}} \cdot \frac{x_{21}}{2 \cdot (b_{0} + b_{1}x_{11} + b_{2}x_{11}^{2})}$$
(19)





Fig. 5a-d. Comparison of static pressure time-series predictions

The 3-variable D-PNN applying extended polynomials (2) of blocks, neurons and square, combination DE terms, tried to predict the static pressure of 1 site locality time-series (Fig.5a-d). It was trained along with the 1-layer ANN the previous day hourly pressure data series (24 or 48 hours, i.e. data samples), which are free on-line available [9]. Meteorological forecasts require as a rule high amount of state input variables to define a complex model, however some tendencies of the progress curves are notable. The D-PNN 2 models double the applied amount of neurons compared with D-PNN 1 (Fig.5.). The DE composition based predictions of the D-PNN seems to succeed any better. The more varied models are formed than ANN (applying 4 or 5 input variables), which is induced by a different neuron combination selection.

#### 6 Conclusion

D-PNN is a new neural network type, which identification and function approximation is based on generalization of data relations. The relative data processing is contrary to common soft-computing method approachs (e.g. ANN), which applications are subjected to a fixed interval of absolute values. This handicap disallows to use various learning and testing data range values (Fig.4.), which may involve real data applications. Thus D-PNN's non-linear regression can cover a generalization of wider interval values. It forms and resolves a DE, composed of sum fractional derivative terms, defining a system model of dependent variables. It is trained only with a small set of input-output data samples, likewise the GMDH algorithm does [1]. The inaccuracies of presented experiments can result from applied incomplete rough training and selective methods, requiring large improvements. Behavior of the presented method differs essentially from other common neural network techniques.

#### Acknowledgement

This work has been elaborated in the framework of the IT4Innovations Centre of Excellence project, reg. no. CZ.1.05/1.1.00/02.0070 supported by Operational Programme 'Research and Development for Innovations' funded by Structural Funds of the European Union and by the Ministry of Industry and Trade of the Czech Republic, under the grant no. FR-TI1/420, and by SGS, VŠB – Technical University of Ostrava, Czech Republic, under the grant No. SP2012/58.

## References

- 1. Das, S., Abraham, A., Konar, A.: Particle swarm optimization and Differential evolution algorithms. Studies in Computational Intelligence (SCI) 116, 1-38, 2008. Springer-Verlag Berlin.
- 2. Iba, H.: Inference of differential equation models by genetic programming. Information Sciences, Volume 178, Issue 23, 1 December 2008, Pages 4453–4468.
- 3. Ivakhnenko, A.G.: Polynomial theory of complex systems. IEEE Transactions on systems, Vol. SMC-1, No.4. 1971.
- 4. Kuneš, J., Vavroch, O., Franta, V.: Essentials of modeling. SNTL Praha 1989 (in Czech).
- 5. Nikolaev, N.Y., Iba, H.: Adaptive Learning of Polynomial Networks. Springer, New York 2006.
- Nikolaev, N. Y., Iba, H.: Polynomial harmonic GMDH learning networks for time series modelling. Neural Networks 16 (2003), 1527–1540. Science Direct.
- Zjavka, L. : Generalization of patterns by identification with polynomial neural network. Journal of Electrical Engineering Vol. 61, No. 2/2010, p. 120-124
- 8. Zjavka, L.: Recognition of Generalized Patterns by a Differential Polynomial Neural Network. Engineering, Technology & Applied Science Research Vol. 2, No 1 (2012).
- 9. National Climatic Data Center of National Oceanic and Atmospheric Administration (NOAA) http://cdo.ncdc.noaa.gov/cdo/3505dat.txt