Neural Networks 73 (2016) 58–69

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Constructing general partial differential equations using polynomial and neural networks

Ladislav Zjavka [a,*], Witold Pedrycz [b,c,d]

[a] VŠB-Technical University of Ostrava, Faculty of Electrical Engineering and Computer Science, Department of Computer Science, 17. listopadu 15/2172 Ostrava, Czech Republic

[b] Department of Electrical & Computer Engineering, University of Alberta, Edmonton T6R 2V4 AB, Canada

[c] Department of Electrical and Computer Engineering, Faculty of Engineering, King Abdulaziz University, Jeddah, 21589, Saudi Arabia

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

ARTICLE INFO

Article history:
Received 9 March 2015
Received in revised form 8 October 2015
Accepted 9 October 2015
Available online 20 October 2015

Keywords:
General partial differential equation composition
Multi-variable function approximation
Differential polynomial neural network
Sum derivative term substitution

ABSTRACT

Sum fraction terms can approximate multi-variable functions on the basis of discrete observations, replacing a partial differential equation definition with polynomial elementary data relation descriptions. Artificial neural networks commonly transform the weighted sum of inputs to describe overall similarity relationships of trained and new testing input patterns. Differential polynomial neural networks form a new class of neural networks, which construct and solve an unknown general partial differential equation of a function of interest with selected substitution relative terms using non-linear multi-variable composite polynomials. The layers of the network generate simple and composite relative substitution terms whose convergent series combinations can describe partial dependent derivative changes of the input variables. This regression is based on trained generalized partial derivative data relations, decomposed into a multi-layer polynomial network structure. The sigmoidal function, commonly used as a nonlinear activation of artificial neurons, may transform some polynomial items together with the parameters with the aim to improve the polynomial derivative term series ability to approximate complicated periodic functions, as simple low order polynomials are not able to fully make up for the complete cycles. The similarity analysis facilitates substitutions for differential equations or can form dimensional units from data samples to describe real-world problems.

1. Introduction

Partial differential equations define a variety of models for function approximation problems of dynamic systems (Rakkiyappan & Dharani, 2015) that are difficult to describe directly by unique explicit expressions (Rakkiyappan, Dharani, & Zhu, 2015). To form the solutions, one can apply evolutionary strategies (Chen, Yang, Meng, Zhao, & Abraham, 2011), genetic programming (GP) (Cao, Kang, Chen, & Yu, 2000; Iba, 2008) or artificial neural networks (ANNs) (Tsoulos, Gavrilis, & Glavas, 2009; Wojciechowski, 2012), which require differential equations to be pre-defined in a general system or standard explicit form. Sum series are widespread used to solve differential equations, e.g. power (Rudd & Ferrari, 2015), Fourier (Chaquet & Carmona, 2012) and polynomial or wavelet series (Liu, Niu, Wang, & Fan, 2014). Polynomial fractions can substitute for differential equation solutions using analogue computation (Bournez, Campagnolo, Graca, & Hainry, 2007). ANN is able to model the non-linear nature of dynamic processes and reproduce a relationship between some inputs and one or more outputs (Rakkiyappan, Zhu, & Chandrasekar, 2014). It can form simple and solid models of systems, the exact solution of whose is problematic or impossible to produce using standard regression techniques (Rakkiyappan, Chandrasekar, Lakshmanan, & Park, 2014). As a rule a lot of soft computing methods utilize some direct composing techniques, e.g. GP or fuzzy models build a required function from a predefined set of operators and terminals to form symbolic tree-like expressions (Cornforth & Lipson, 2013). In the same vein, the common ANNs cannot create more complex models using only flat 1 or 2-layer structures; they apply only absolute values of input variables, which are not able to describe a wider data range than specified by the training samples. Thus the ANN generalization abilities formed on a basis of the training data may be difficult

* Corresponding author.
E-mail addresses: lzjavka@gmail.com (L. Zjavka), wpedrycz@ualberta.ca (W. Pedrycz).

http://dx.doi.org/10.1016/j.neunet.2015.10.001
0893-6080/© 2015 Elsevier Ltd. All rights reserved.

or problematic if the model has been trained with inputs or outputs that are quite far from those forming the testing data (Giles, 2001). A trial could be made in looking at a vector of input variables as a no compact "pattern" but a bound dependent point set in the N-dimensional space. Artificial neurons can apply multi-variable polynomials to generalize from observations partial data relations points into some convergent substitution sum series of partial relative derivative terms of the composite function model. In contrast with the ANN functionality, each neuron (i.e., the derivative term) output may be directly included in the combination sum of the total network output calculation. Biological neural cells seem to follow similar principles. The dendrites collect signals incoming from other neurons but unlike the ANN functionality the signals can interact already in single branches. Multi-variable polynomials can ease model this framework by means of multiplying (products) some input variables. The weighted combinations are summed in the cell of the body and then transformed through a time-pulse dynamic periodic activation function (the activated neural cell generates series of time-delayed output pulses in response to its input signals) (Benuskova, 2002). The period of this activation function depends on some non-linear combinations of input variables and seems to represent a derivative part of a single differential equation term substitution. According to these assumptions, brain applies combined techniques of relative data processing and dynamic periodic functions to form derivative terms substitutions of systems of differential equations in time-dependent pulse models, very efficient for a large scale variability, variable-differences and adaptability of varied-shape input pattern forms.

Differential polynomial neural network (D-PNN) is a new neural network type, which extends a complete multi-layer polynomial neural network (PNN) structure to produce substitution relative derivative terms, which selected combination can define and solve an unknown general partial differential equation of a searched multi-variable function model. The D-PNN forms its functional output as a generalization of partial polynomial data relations however it can also decompose a visual input pattern into some characteristic matrix fragments to model the elementary point (feature) relations, possible to identify by the searched function (analogous to the ANN function approximation and pattern recognition). This way the D-PNN relative derivative model can correctly classify any untrained variable-shape pattern forms, which keeps the trained data relations (modelled function), regardless of the size and position in the input matrix (Zjavka, 2012b). A number of technical, biological, and psychological studies suggest the brain applies just relative forms of input variables (e.g. contrary to the absolute signals of a CCD camera, which must be comparatively processed in a solar image Druckmüller, 2009) and a reductive decomposition of complete input patterns into some major characteristic elements (low-level properties) (Fiset et al., 2008). Line terminations are by far the most important features for the correct human letter identification. Other features as intersections, curvatures or slants are considered to a lesser extent (Willenbockel et al., 2010). Generalized relations of the elementary fragment positions might define a variable-shape (size-independent) type model for all visual pattern forms (Zjavka, 2012b).

The D-PNN relative models are able to apply quite different training and testing intervals of input and output data values. The particle swarm optimization, usually used for a continuous optimization (Abraham, Guo, & Liu, 2006), can cope with binary combinatorial problems (Yuan, Nie, Su, Wanga, & Yuan, 2009) like the derivative term combination selection from the substitution sum series in a differential equation solution. Sections 2 and 3 describe the theoretical background of the general derivative model and provide math proofs of the simplest linear form validity of the derivative term fraction substitutions. Section 4 presents the D-PNN multi-layer architecture and its principles used for the substitution and calculation of composite derivative terms. Section 5 presents test approximations of several types of periodic multi-variable benchmarks and comparisons of the D-PNN and ANN models mostly on the complete trained intervals of matrix point hyper-surfaces of 3-dimensional functions. Section 6 tests two different types of neural networks to model real multi-variable functions, which some fluctuant unstable weather data relations may represent. All the presented benchmark and real data models apply only a basic PNN architecture with a limited number of input variables, which do not increase the number of combination couples in each hidden layer and it is not necessary to select some of them. The PNN application in the formation and solution of the general differential equation is a novelty in this field, however the experimental results indicate that the method is efficient (using only a few derivative terms Zjavka & Abraham, 2013) and can model non-linearities of physical or natural dynamic processes and systems, which general differential equations may conveniently describe and which are too uncertain or complex to be described by means of exact computational techniques. A standard notation used in all formulas, is outlined below.

x_i—input variables or particle value/state
y_i—polynomial or substitution term output, Y—overall (network) output
u—modelled function, $f(y)$—composite function
a, b—polynomial parameters,
w_i—weights of terms
sig—sigmoidal function, rbf—radial basis function
neuron—substituting fraction DE term, CT—Composite Terms (using composite functions derivatives)
P_A—probability of activation of neurons
DE—Differential Equation
PNN—Polynomial Neural Network
ANN—Artificial Neural Network
RMSE—Root Mean Squared Error

2. Theoretical background of the partial derivative terms substitution

The D-PNN uses a complete PNN structure to produce substitution polynomial fraction sum terms that can solve the general partial differential equation according to the similarity dimensional analysis, which forms system characteristics from dimensionless ratio groups of variables (Price, 2003). The analysis leads to a set of independent dimensionless factors, which may carry information about the behaviour of a system and represent the major reduced variables, applied to the regression function in place of the original measurements. This scale-invariance model can replace the standard data transformations (Randall, 2012). The Buckingham π-theorem proves the original unknown relationship, represented by $f(x_1, x_2, \dots, x_n) = 0$, where x_i are the variables, can be transformed into a new function $\phi(\pi_1, \pi_2, \dots, \pi_{n-m})$ of $n - m$ independent dimensionless reduced products π_j of the original x_i variables, where m is the number of fundamental dimensions out of which the dimensions of the original variables are composed (Vignaux, 1992). The dimensional analysis is the most useful in the case that a mathematical model is not known however it can investigate a differential equation, which describes a physical process or principle. For instance, a motion of a simple pendulum (1) can be approximated for restricted values of the initial conditions and the angle ϕ_0 by means of a linear model (2).

$$\frac{d^2\phi}{dt^2} = -\frac{g}{L}\sin\phi \qquad (1)$$

ϕ—angle of the line