Information Sciences 281 (2014) 462–477

Contents lists available at ScienceDirect

Information Sciences

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

Constructing ordinary sum differential equations using polynomial networks

Ladislav Zjavka *, Václav Snášel

VŠB-Technical University of Ostrava, IT4innovations Ostrava, Czech Republic

ARTICLE INFO

Article history:
Received 24 May 2013
Received in revised form 7 April 2014
Accepted 20 May 2014
Available online 29 May 2014

Keywords:
Polynomial neural network
Differential equation composition
Sum relative derivative term substitution
Multi-parametric function approximation

ABSTRACT

Data relations can define general sum partial differential equations of a composite function additive derivative model. Time-series data observations can analogously describe an ordinary sum differential equation with time derivatives, which is possible to be solved using partial derivative term substitutions of time-dependent series. Differential polynomial neural network is a new type of neural network, which constructs and substitutes for an unknown general partial differential equation from data observations, developed by the author. It generates sum series of convergent partial polynomial derivative terms, which can describe an unknown complex function time-series. This type of non-linear regression decomposes a system model, described by the general differential equation, into many partial low order derivative specifications of selected relative sum terms. Common soft-computing techniques in general can apply input variables of only absolute interval values of a specific data range. The character of relative data allows processing a wider range of test interval values than defined by a training set. The characteristics of the composite sum differential equation solutions can facilitate a much greater variety of model forms than is allowed using standard soft computing methods. Recurrent neural network proved to form simple solid time-series models, most of which it is possible to describe using ordinary differential equations, so the comparisons were done.

1. Introduction

Differential equations can describe a lot of complex systems, which it is difficult to model by unique exact functions. The solutions can apply sum series [3,7], genetic programming (GP) [5,9,14], fuzzy techniques [2] or evolutionary algorithms [8] and an artificial neural network (ANN) construction [24], which require the differential equation to be defined in an explicit form. ANN is able to model the non-linear nature of dynamic processes and reproduce an empirical relationship between some inputs and one or more outputs. It can define simple and solid models, the exact solution of which is problematic or impossible to get using standard regression techniques. Several methods were developed, where ANN models can substitute for predetermined differential equations. The initial/boundary conditions are defined with no adjustable parameters, while a multilayer ANN can approximate any function derivatives. The sigmoidal [13] or radial basis [16] activation functions of neurons can realize the derivatives of a searched function using an appropriate network topology. Each following hidden layer commonly increases the derivative order of the applied activation function according to the differential

* Corresponding author. Tel./fax: +420 59732 5252.
E-mail addresses: lzjavka@gmail.com (L. Zjavka), vaclav.snasel@vsb.cz (V. Snášel).

http://dx.doi.org/10.1016/j.ins.2014.05.036
0020-0255/

equation definition [25]; this way the neural network can form kth derivatives of the searched function in the kth layer. Another method is based on the fact that single hidden layer feed-forward networks with a linear single output layer are capable of approximating arbitrary functions and its derivatives. It can compose directly a differential equation solution, considering boundary and/or initial conditions, from several ANNs, which neurons apply the competent derivatives of activation functions to form corresponding derivative terms [1]. A common ANN operating principle is based on learned entire similarity relationships between new presented input patterns and the trained ones; however, it does not allow for eventual straight elementary data relations, which multi-parametric polynomial functions can easily describe. The ANN generalization from the training data set may be difficult or problematic if the model has not been trained with inputs around the range covered testing data [12]. Standard soft-computing techniques utilize direct computational methods, which can operate only within the range of absolute interval values of input variables, e.g. GP or fuzzy models compose a searched function using collections of operators and terminals from a defined set to form symbolic expressions [18]. If data involve relations, which may become stronger or weaker character, the network relative model could generalize it into wide-range valid values. It might combine a neural network composite function formation with differential equation solutions of substitute sum series. This way a general derivative model is decomposed into series of partial relative composite function descriptions. Recurrent neural network (RNN) is often used to define power-full models of time-series data samples, most of which is possible to describe using ordinary differential equations. It applies as inputs also its neuron outputs from a previous time estimate. Analogous to other common neural network solutions, it is not possible to get the specifications of models in the form of a mathematical description which can, if necessary, be improved using other computational methods; they seem to their users to be a "black box".

$$Y = a_0 + \sum_{i=1}^{m} a_i x_i + \sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij} x_i x_j + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} a_{ijk} x_i x_j x_k + \cdots \tag{1}$$

m – number of variables $X(x_1, x_2, \ldots, x_m)$ $A(a_1, a_2, \ldots, a_m), \ldots$ – vectors of parameters

Differential polynomial neural network (D-PNN) is a new neural network type, which results from the GMDH (Group Method of Data Handling) polynomial neural network (PNN), created by a Ukrainian scientist Aleksey Ivakhnenko in 1968, when the back-propagation technique was not known yet [15]. It is possible to express a general connection between input and output variables by means of the Volterra functional series, a discrete analogue of which is the Kolmogorov–Gabor polynomial (1). This polynomial can approximate any stationary random sequence of observations and can be computed by either adaptive methods or a system of Gaussian normal equations. GMDH decomposes the complexity of the process into many simpler relationships each described by low order polynomials (2) for every pair of input values x_i, x_j. It defines the optimal structure of complex system model with identifying non-linear relations between input and output variables. Typical GMDH network maps a vector input $\boldsymbol{x}$ to a scalar output y, which is an estimate of the true function $f(\boldsymbol{x}) = y^t$ [20]. PNN can apply a self-organizing structure [21], several types of evolutionary, genetic or iterative algorithms [22] and polynomial transfer functions [10].

$$y = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j + a_4 x_i^2 + a_5 x_j^2 \tag{2}$$

D-PNN forms and resolves an unknown partial differential equation (DE) of a searched function description from real data observations. A general DE is substituted producing sum series of selected fractional polynomial derivative terms, decomposing the system model into many partial derivative relations. This adjective series model type is different from simple upright compositing computational techniques, which can compose a searched function using a collection of operators and terminals of a defined set to form symbolic tree-like structural expressions. In contrast with the ANN functionality, each neuron (i.e. DE term) can take part directly in the total network output sum calculation. D-PNN block skeleton is formed by the GMDH polynomial network; however, its operating and construction principles differ from those of GMDH, being based on Taylor-series expansions, as the polynomial power degrees double each following layer addition. It extends the basic complete GMDH network structure to generate sum derivative term series [26]. The application of PNN to solve a (general) DE is novelty, however the experimental results indicate the method is efficient even using only several neurons [27] and can contribute to the field. It should approve mainly in the modeling of physical and natural dynamic systems, which are possible (but not exactly) to describe by differential equations and which are too complex (or hardly) to be solved unambiguously using standard soft computing techniques. Analogous to the GMDH decomposition of the Kolmogorov–Gabor polynomial (1), a general DE is decomposed by a composite function backward network model using the GMDH polynomial (2) and its variables into the 2nd order partial DE term descriptions of an additive solution.

2. General sum differential equation composition

D-PNN forms and resolves a partial DE (3), in which an exact definition is not known in advance and can generally describe a system model of dependent variables, using summation derivative terms. The searched function u may be expressed in the form of sum series (4), consisting of series arising from derivative sum convergent term series (5) in the case of 2 input variables. The study tried to replace the sum partial DE (3) with series of fractional multi-parametric polynomial terms (8), which can describe relative derivative dependent changes over some combinations of variables of a DE solution.