Expert Systems with Applications 40 (2013) 4528–4538

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

Optimization of self-organizing polynomial neural networks

Ivan Maric *

Rudjer Boskovic Institute, Bijenicka 54, 10000 Zagreb, Croatia

ARTICLE INFO

Keywords:
Polynomial neural networks
GMDH
Levenberg–Marquardt algorithm
Particle swarm optimization
Time series modeling

ABSTRACT

The main disadvantage of self-organizing polynomial neural networks (SOPNN) automatically structured and trained by the group method of data handling (GMDH) algorithm is a partial optimization of model weights as the GMDH algorithm optimizes only the weights of the topmost (output) node. In order to estimate to what extent the approximation accuracy of the obtained model can be improved the particle swarm optimization (PSO) has been used for the optimization of weights of all node-polynomials. Since the PSO is generally computationally expensive and time consuming a more efficient Levenberg–Marquardt (LM) algorithm is adapted for the optimization of the SOPNN. After it has been optimized by the LM algorithm the SOPNN outperformed the corresponding models based on artificial neural networks (ANN) and support vector method (SVM). The research is based on the meta-modeling of the thermodynamic effects in fluid flow measurements with time-constraints. The outstanding characteristics of the optimized SOPNN models are also demonstrated in learning the recurrence relations of multiple superimposed oscillations (MSO).

1. Introduction

Approximation of complex multidimensional systems by SOPNN, also known as the GMDH polynomial neural networks (PNN), was introduced by Ivakhnenko (1971). The SOPNN are constructed by combining the low order polynomials into multi layered polynomial structures where the coefficients of the low-order polynomials (generally 2-dimensional 2^{nd} order polynomials) are obtained by polynomial regression (Chapra & Canale, 1998) with the aim to minimize the approximation error. GMDH models may achieve reasonable approximation accuracy at low complexity and are simple to implement in digital computers (Maric & Ivek, 2011). The GMDH is resistant to over-fitting since it uses separate data sets for regression and for model selection. When applied to real time compensation of nonlinear behavior, the self-organizing nature of GMDH may eliminate the complicated structural modeling and parameterization, common to conventional modeling strategies (Iwasaki, Takei, & Matsui, 2003).

The performance of the SOPNN is generally evaluated by a single parameter measure (Witten & Eibe, 2005), typically by the least square error, which minimizes the model approximation error rather than its complexity. When building the models for time-constrained applications the constraints can be efficiently embedded into the model selection metrics (Maric & Ivek, 2011). It was shown (Maric & Ivek, 2011) that the raw SOPNN models (GMDH PNN) are inferior to multilayer perceptron (MLP) when considering the accuracy with respect to the complexity. It was also concluded (Maric & Ivek, 2011) that SVM is not appropriate for building the low complexity models for time-constrained applications.

The SOPNN node-polynomial weights, after calculated by the regression, remain unchanged during the rest of the training process resulting in sub-optimal SOPNN models. The accuracy and the prediction of models may be improved significantly when trained by the genetic programing and back-propagation (BP). It was shown (Nikolaev & Iba, 2003) that the population-based search technique, relying on the genetic programing and the BP algorithm, enables to identify the networks with good training as well as generalization performances. The BP improves the accuracy of the model but it is known to often get stuck in local minima. The idea of this paper is to adapt a more robust procedure for the optimization of the SOPNN relation with respect to its weights.

The PSO is a nature inspired algorithm, which enables the optimization of model weights by simulating the flight of a bird flock (Eberhart & Kennedy, 1995). Since the PSO is simple to implement it has been used in our experiments for the estimation of the approximation abilities of raw SOPNN models. After the PSO improved significantly the approximation accuracy of various SOPNN models a more complex Levenberg–Marquardt (LM) algorithm (Levenberg, 1944; Marquardt, 1963) has been adapted for the optimization of model weights. Although widely used for the optimization of ANN, the use of the LM algorithm for the optimization of weights of the SOPNN to the best of my knowledge has not been reported in the literature. The LM algorithm converges many times faster than the PSO and increases the approximation accuracy of the SOPNN model substantially. This paper describes the adapta-

* Tel.: +385 1 4561191.
E-mail address: ivan.maric@irb.hr

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.01.060

tion of PSO and LM algorithm for the optimization of SOPNN and demonstrates how the approximation accuracy of the original GMDH model can be significantly improved after optimizing its weights.

In the following section, the GMDH, PSO and the LM algorithm are described. The PSO and the LM algorithm are adapted for the optimization of SOPNN weights. Section 3 describes the procedure for the estimation of the execution time for SOPNN and MLP in time constrained applications. A procedure for the compensation of thermodynamic effects in flow rate measurements is summarized in section 4 and the results of the simulations of the flow rate error compensation procedure by the surrogate models are given in section 5. Finally in section 6, the outstanding performances of the SOPNN are demonstrated on MSO task that has been widely studied in echo state networks (ESN) literature.

2. GMDH, PSO and LM algorithm

2.1. GMDH algorithm

The GMDH algorithm (Ivakhnenko, 1971) constructs the models by combining the low-order polynomials into multi layered polynomial networks. Fig. 1 illustrates a complete two-layer feed-forward SOPNN representing a 3-dimensional system, where $p_{\lambda,i}$ denotes a low-order and low-dimensional polynomial corresponding to the ith node of the layer λ and x_i represents the ith independent variable.

First order and second order two-dimensional polynomials are preferred as their cascading does not increase rapidly the order of overall polynomial relation and because they are fast to process. In this paper we will restrict our attention to a complete second-order two-dimensional polynomial

$$p_{\lambda i} = a_{\lambda i1} + a_{\lambda i2} z_{\lambda i1} + a_{\lambda i3} z_{\lambda i2} + a_{\lambda i4} z_{\lambda i1}^2 + a_{\lambda i5} z_{\lambda i2}^2 + a_{\lambda i6} z_{\lambda i1} z_{\lambda i2}, \quad (1)$$

where $z_{\lambda i1}$ and $z_{\lambda i2}$ represent the input variables and $a_{\lambda i1},\ldots,a_{\lambda i6}$ are the corresponding weights (coefficients) obtained by the polynomial regression. Note that $z_{\lambda i1}$ and $z_{\lambda i2}$ can be any combination of two different variables from lower layers including the independent variables (x_i) and the derived regression polynomials ($p_{\lambda i}$). For example, the input variables for the polynomial $p_{2,6}$ from Fig. 1 are the polynomial $p_{1,1}$ from the first layer and the independent variable x_3.

The GMDH algorithm assumes two independent data sets: A training set of M samples

$$\mathcal{D}_t = \{(\mathbf{x}_{ti}, y_{ti})\}_{i=1}^{M}, \quad (2)$$

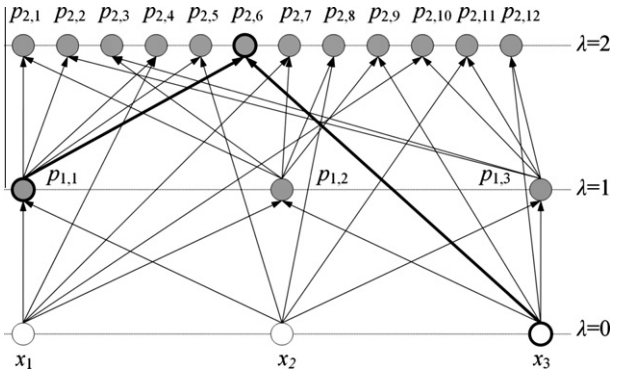

Fig. 1. Illustration of SOPNN construction.

where each sample consists of a data vector $\mathbf{x}_{ti} = (x_{ti1}, x_{ti2},\ldots,x_{tiK})$, $x_{ti} \in \mathcal{R}^K$, and the corresponding dependent variable $y_{ti} \in R$, as well a validation data set of N samples

$$\mathcal{D}_v = \{(\mathbf{x}_{vi}, y_{vi})\}_{i=1}^{N}, \quad (3)$$

with data vector $\mathbf{x}_{vi} = (x_{vi1},x_{vi2},\ldots,x_{viK})$, $x_{vi} \in \mathcal{R}^K$, and the corresponding dependent variable $y_{vi} \in \mathcal{R}$. The algorithm uses the training data set (2) to fit the coefficients of the regression polynomial (1) and the validation set (3) to verify the approximation error of the polynomial. The polynomials are then ranked according to some predefined metrics (Maric & Ivek, 2011).

In order to calculate the coefficients, $a_{\lambda i1},\ldots, a_{\lambda i6}$, in (1) by the polynomial regression, a set of 6 simultaneous linear equations

$$\left\{\frac{\partial}{\partial a_{\lambda ik}} \sum_{m=1}^{M} \left(y_{tm} - p_{\lambda i}^{tm}\right)^2 = 0\right\}_{k=1}^{6} \quad (4)$$

must be solved, where M is a total number of training samples, y_{tm} is the mth sample value of the dependent variable from the training data set (2) and $p_{\lambda i}^{tm}$ is the value of the ith polynomial at layer λ corresponding to the mth data vector from the training data set (t). In our implementation of the GMDH algorithm the above set of linear equations is solved by the Gauss elimination method using forward elimination, back substitution, and pivoting (Chapra & Canale, 1998).

As illustrated in Fig. 1. the total number of possible nodes in each layer is increasing rapidly by the increase of the layer number and can be easily calculated by the following simple iterative equation $N_{\lambda+1} = \frac{N_\lambda \cdot (N_\lambda - 1)}{2} + N_\lambda \sum_{i=0}^{\lambda-1} N_i$, where $\lambda = 0,1,\ldots$ denotes the corresponding layer, N_λ is the total number of nodes at layer λ and the second term in the equation equals zero for $\lambda = 0$. To prevent the combinatorial explosion the maximum number of nodes retained per layer is generally limited. The nodes retained at lower layers are combined multiple times to produce the nodes at higher layers. To speedup the algorithm the retained polynomials may be tabulated.

2.2. Particle swarm optimization of SOPNN weights

Let $\mathbf{a} \in \Re^N$ be the vector, $\{a_i\}$, $i = 1,\ldots, N$, in N-dimensional space. Let the SOPNN polynomial relation $P(\mathbf{x},\mathbf{a})$ be the particle whose position in the space is defined by the vector $\mathbf{a}$. Particle swarm optimization (Eberhart & Kennedy, 1995) minimizes the nonlinear fitness function:

$$e(\mathbf{a}) = \sum_{k=1}^{K} [y_k - P(\mathbf{x}_k, \mathbf{a})]^2, \quad (5)$$

where K is the total number of training vectors, $\mathbf{x}_k$ denotes the kth M-dimensional input training vector, $y_k = y(\mathbf{x}_k)$ is the corresponding training value and $\mathbf{a}$ denotes N-dimensional vector representing the coefficients of all nodes of the SOPNN polynomial $P(\mathbf{x}_k,\mathbf{a})$. Before starting the PSO the position $\mathbf{a}$ and the velocity $\mathbf{v}$ of each particle are initialized by:

$$a_{ki}^0 = L_i + r_{ki}|U_i - L_i|, \quad k = 1,\ldots,K, \quad i = 1,\ldots,N \quad (6)$$

and

$$v_{ki}^0 = (2s_{ki} - 1) \cdot |U_i - L_i|, \quad k = 1,\ldots,K, \quad i = 1,\ldots,N, \quad (7)$$

where a_{ki}^0 and v_{ki}^0 are the corresponding ith component of the kth particle's initial position and velocity, L_i and U_i are the corresponding lower and upper boundaries for the ith dimension of the search space, and r_{ki} and s_{ki} denote the random numbers from the range [0,1]. For each particle k, the best particle position A_k and the best