Expert Systems with Applications 36 (2009) 7416–7423

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Multi-objective optimization of TSK fuzzy models

O. Guenounou [a,*], A. Belmehdi [a], B. Dahhou [b]

[a] Laboratory of Industrial Technology and Information LT2I, University of Bejaia, Route de l'université Targa Ouzemour, Béjaia, Algeria
[b] Laboratory for the Analysis and Architecture of Systems LAAS, Toulouse, France

ARTICLE INFO

Keywords:
Backpropagation
Genetic algorithms/NSGA-II
Fuzzy rules
Hybrid algorithm
Structure

ABSTRACT

In this paper we propose a hybrid algorithm to optimize the structure of TSK type fuzzy model using backpropagation (BP) learning algorithm and non-dominated sorting genetic algorithm (NSGA-II). In a first step, BP algorithm is used to optimize the parameters of the model (parameters of membership functions and fuzzy rules). NSGA-II is used in a second phase, to optimize the number of fuzzy rules and to fine tune the parameters. A well known benchmark is used to evaluate performances of the proposed modelling approach, and compare it with other modelling approaches.

1. Introduction

Recently, intelligent systems including fuzzy systems (FS), neural networks (NN) and genetic algorithms (GA) have been shown to be an important tool for modelling and control of complex systems (Chang, Yu, & Yu, 2004; Tan, Lun, Loh, & Tan, 2005; Wang & Langari, 1996).

The idea to combine between these techniques, has attracted the attention of several researchers of which the goal is to put at common the advantages of each technique to design intelligent systems, autonomous, fast and able to control complex systems without knowing their mathematical models.

The two usually combinations are NN-FS and GA-FS. The first hybridization approach (NN-FS) is a realization of the functionality of fuzzy systems using neural networks where the connection weights of the neural network correspond to the parameters of the fuzzy systems. The aim of this approach is to provide a design method of fuzzy systems using learning algorithms derived from the neural network paradigm like backpropagation (BP) algorithm. This algorithm (Narendra & Parthasarathy, 1991) remains the more used in training neuro fuzzy systems (NFS), although the suggested architectures are different (Jang, 1993; Lee & Teng, 2000).

The second hybridization approach (GA-FS), consists in the design of fuzzy systems using genetic algorithms. Karr (Karr & Gentry, 1993) has successfully used these algorithms to optimize only the membership functions parameters for a pH control process (fuzzy rules being already build), while other authors (Belarbi, Titela, Bourebiaa, & Benmahammed, 2005; Hsu & Yeh, 1994), have used them to design control rules for fuzzy controller. Recently, hierarchical genetic algorithms-HGA (an improved version of standard genetic algorithms) are also applied. In Acosta and Todorovich (2003), HGA are used to search the minimal number of membership functions to associate with each variable of fuzzy controller whereas in Guenounou, Belmehdi, and Dahhou (2007) they are used to design a fuzzy controller with minimal number of fuzzy rules. In this paper, we propose a new approach for modelling complex dynamical systems using intelligent techniques. The model is a feedforward neural fuzzy model (NFM) with Takagi–Sugeno fuzzy rules (Takagi & Sugeno, 1985). The optimization of the structure of NFM including parametric optimization is offered by the proposed hybrid algorithm which is performed into two phases. In the beginning, we use the BP algorithm to optimize the parameters related to membership functions and fuzzy rules. In the second phase, we use a multi-objective genetic algorithm, called NSGA-II (non-dominated sorting genetic algorithm II) to a fine-tuning of the parameters and to reduce the number of fuzzy rules. This method aims to combine the advantages of each algorithm and to remedy their drawbacks. The backpropagation algorithm is fast and precise; however it manages badly the local optima. For that, the results obtained by this algorithm, in the first phase, constitute a starting point for the NSGA-II which acts in the second phase for a global optimization. This paper is organized as follows. Section 2 shows the structure of the used neural fuzzy model. Section 3 gives a brief overview of NSGA-II. Section 4 describes the proposed hybrid learning algorithm. Section 5 presents the simulation results. Finally in Section 6, a general conclusion is given.

2. Neural fuzzy model structure

In this section, the structure of the neural fuzzy model (NFM) is presented (Fig. 1). The network is divided into five layers. A system with r input variables $\vec{x} = [x_1, x_2 \ldots, x_r]$ and a single output y is considered here for convenience. Accordingly, there are r nodes in layer 1 and one node in layer 5. Nodes in layer 1 are input nodes. Nodes in layer 2 are term nodes that act as membership function

* Corresponding author.
E-mail address: wguenounou@yahoo.fr (O. Guenounou).

0957-4174/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2008.09.044

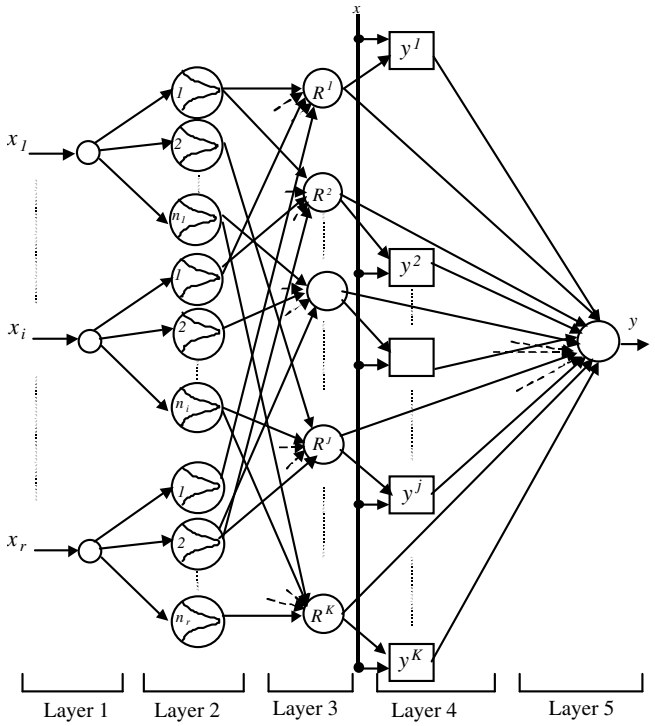

Fig. 1. Neural fuzzy model structure.

representing the fuzzy labels of the input in their universes of discourse. Each node in layer 3 is a rule node that represents one fuzzy rule. All nodes in layer 3 form the fuzzy rule base of the NFM. The links of this layer define the precondition of fuzzy rules. For each rule node there are r fixed links from the term nodes of layer 2. Nodes in layer 4 are called consequent nodes. Each node in this layer performs the consequent of a fuzzy rule that it represents. In this work, we adopt the TSK-type neural fuzzy model and an arbitrary fuzzy rule R^i of the following form:

$$R^i: \text{if } x_1 \text{ is } A_1^i \text{ and } x_2 \text{ is } A_2^i \ldots \text{ and } x_r \text{ is } A_r^i \text{ then } y^i = a_0^i + a_1^i \cdot x_1 + a_2^i \cdot x_2 \ldots + a_r^i \cdot x_r \quad (1)$$

where $A_1^i, A_2^i \ldots$ and A_r^i are fuzzy sets (usually corresponding to linguistic labels, such as negative (N), zero (Z) and positive (P)) and y^i is the output (consequent) of fuzzy rule R^i.

In layer 5 the node is called a defuzzification node.

To give a clear description, we will detail the functionality of NFM layer by layer. For notation convenience, the input to the ith node in layer k is denoted by u_i^k and the output value by O_i^k.

Layer 1: The nodes of this layer directly transmit input signals to the next layer. That is

$$O_i^1 = u_i^1, \quad u_i^1 = x_i \quad (2)$$

Layer 2: Every node i in this layer is an adaptive node with an output node defined by

$$O_i^2 = \exp\left(-\frac{(u_i^2 - c_i)^2}{2\sigma_i^2}\right) \quad (3)$$

$$u_i^2 = \begin{cases} O_1^1 & \text{if } i = 1, 2, \ldots, n_1 \\ O_2^1 & \text{if } i = n_1 + 1, \ldots, n_1 + n_2 \\ O_j^1 & \text{if } i = n_1 + \cdots + n_{j-1} + 1, \ldots, n_1 + \cdots + n_j \\ \cdot & \\ O_r^1 & \text{if } i = n_1 + \cdots + n_{r-1} + 1, \ldots, n_1 + \cdots + n_r \end{cases} \quad (4)$$

where $n_1, n_2, \ldots$ and n_r are the number of fuzzy sets associated with the input variables $x_1, x_2, \ldots$ and x_r, respectively. c_i and σ_i are, respectively, the center and the width of the Gaussian membership function (Matlab fuzzy-logic toolbox) of the ith term node. Note that layer 2 links are all set to unity.

Layer 3: Each node in this layer corresponds to a fuzzy rule in the rule base unit. Its inputs come from all nodes in layer 2 which participate in the premises part of that rule. The output of each node represents the firing strength of a rule and is calculated via the product operation:

$$O_i^3 = \prod_j u_j^3, u_j^3 = O_j^2, i = 1, 2 \cdots K \quad (5)$$

where K is the number of fuzzy rules, equal to $n_1 \times n_2 \cdots \times n_r$. The link weights in this layer are also set to unity.

Layer 4: Each node in this layer performs a linear summation of the input variables as shown by the following equation:

$$O_i^4 = a_0^i + \sum_{j=1}^{r} a_j^i \times x_j = y^i \quad (6)$$

where $a_j^i (j = 0, 1, \ldots r$ and $i = 1, 2, \ldots K)$ are the parameters to be tuned. Links from this layer to output layer are all equal to unity.

Layer 5: The node in this layer is employed to calculate the output signal of the NFM. The output node is given by the following equation:

$$O^5 = \frac{\sum_{j=1}^{K} O_j^3 O_j^4}{\sum_{j=1}^{K} O_j^3} = y \quad (7)$$

3. General principle of NSGA-II

The genetic algorithm, used in this paper, is an adaptation of a general structure of multi-objective genetic algorithms, called NSGA-II (non-dominated sorting genetic algorithm II) recently proposed by Deb, Pratap, Agarwal, and Meyarivan (2002). This algorithm provides excellent results compared to others proposed multi-objective genetic algorithms such as its first version (Srinivas & Deb, 1995). In this section, we present the general principle of NSGA-II, using the same notations adopted in Deb et al. (2002). Then we specify its components, for our problem, in Section 4. NSGA-II algorithm uses an elitist approach which can significantly speed up the performance of GA (Meunier, Talbi, & Reininger, 2000; Zitzler, Deb, & Thiele, 2000) a sorting procedure, based on a fast algorithm and a comparison operator based on calculus of a crowding distance instead a sharing function (Horn, Nafpliotis, & Goldberg, 1994) which has been proved to be problematic (depends largely on the chosen sigma value). The general principle of NSGA-II is presented in Fig. 2: at each generation t, a parent population P_t of size N and an offspring population Q_t of the same size are merged for forming a population $R_t (R_t = P_t \bigcup Q_t)$ of size $2N$. Then, the population R_t is partitioned into a number of sets called fronts F, which are constructed iteratively. Front F_1 consists of the non-dominated solutions from R_t. F_2 consists of non-dominated solution from the set $(R_t - F_1)$ and so on. In general, F_i consists of the non-dominated solutions from the set $(R_t - (F_1 \bigcup F_2 \ldots \bigcup F_{i-1}))$. Deb et al. (2002) have proposed a fast partitioned algorithm called fast non-dominated sorting algorithm. Once all fronts are identified, a new parent population P_{t+1} of size N is formed by adding the fronts to P_{t+1} in order (front one F_1 followed by front two and so on) as long as the size of R_t do not exceed N individuals. If the number of individuals present in P_{t+1} is lower than N, a crowding procedure is applied to the first front F_i not included in P_{t+1}. The aim of this procedure is to insert the $N - P_{t+1}$ best individuals which miss to fill all population R_t. For each solution i in F_i a crowding distance d_i is calculated based on each objec-