Neurocomputing 129 (2014) 458–466

Contents lists available at ScienceDirect

Neurocomputing

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

Novel calculation of fuzzy exponent in the sigmoid functions for fuzzy neural networks

János Botzheim [a,b,*], Péter Földesi [c]

[a] Graduate School of System Design, Tokyo Metropolitan University, 6-6 Asahigaoka, Hino, Tokyo 191-0065, Japan
[b] Department of Automation, Széchenyi István University, 1 Egyetem tér, Győr 9026, Hungary
[c] Department of Logistics and Forwarding, Széchenyi István University, 1 Egyetem tér, Győr 9026, Hungary

ARTICLE INFO

Article history:
Received 2 December 2012
Received in revised form
11 July 2013
Accepted 30 September 2013
Communicated by Ligang Wu
Available online 18 October 2013

Keywords:
Fuzzy neural network
Sigmoid function
Fuzzy exponent

ABSTRACT

This paper presents a novel calculation of fuzzy exponent in the sigmoid functions for fuzzy neural networks. The investigated fuzzy neural network applies fuzzy input signals and crisp connection weights in the network's hidden and output layers. The applied calculation of fuzzy exponent is based on a parametric representation of the fuzzy exponent that is able to provide a crisp output instead of the extension principle's fuzzy output and requires significantly less computational effort than the learning based on α-cuts. For the training of the network the bacterial memetic algorithm is applied which effectively combines the bacterial evolutionary algorithm with gradient based learning. The method is tested on a benchmark problem and on two real datasets. Comparison to the classical technique concerning the learning time is also provided in the paper.

1. Introduction

Fuzzy neural networks (FNN) are usually classified as three main types [1]:

- FNN based on fuzzy operators.
- Fuzzified neural networks.
- Fuzzy inference networks.

In this paper we deal with fuzzified neural networks. One class of fuzzified neural networks is regular FNN whose inputs, outputs and connection weights are fuzzy sets, its topological architecture is identical to a crisp neural network, and the internal operations are based on Zadeh's extension principle [2,3] and fuzzy arithmetic.

Regular fuzzy neural networks can be classified into three categories according to the fuzziness of the inputs and connection weights [1]. The first one includes FNNs with crisp inputs and fuzzy weights. The second category contains FNNs with fuzzy inputs and crisp weights. The third category is the most general with fuzzy inputs and fuzzy connection weights.

In this paper we deal with the second category where the inputs are fuzzy and the weights are crisp. However, we propose some modifications on regular FNNs: we use a topological architecture which is identical to a crisp neural network, we use fuzzy inputs and crisp weights, but the internal operations are not based on Zadeh's extension principle in our case. Regardless of the actually used training method, the calculation of fuzzy exponent is a time consuming process, and the novel calculation based on a parametric representation [4] can reduce the running time significantly, and it is shown in [4] that it approximates the result obtained by the extension principle. It gives a crisp output of a fuzzy input, thus, the output of the FNN will be crisp instead of fuzzy as also suggested in [5]. Practical applications, examples are presented in Section 4.

Fuzzy neural networks have many applications [1,6,7], thus their training algorithms are worth improving. Amongst others evolutionary algorithms and gradient based techniques are very popular FNN training methods [8,9]. The problem with the evolutionary algorithms is that although they can perform global optimization of the parameters, their convergence to the optimum is slow. The gradient based methods have faster convergence, but in many cases they converge to a local optimum. Another drawback of the gradient based techniques is that they need derivative calculation which can be a very time consuming task if the calculation has to be performed several times. In case of fuzzy numbers, the classical methods need to perform the derivative calculations on lots of different α-cuts [10]. In order to speed up the process a parametric representation is used, so the calculation has to be done only on one α-cut.

To demonstrate the effectiveness of the proposed novel calculation of fuzzy exponent in the sigmoid functions for FNN, the bacterial memetic algorithm [11] is proposed for the training.

* Corresponding author at: Graduate School of System Design, Tokyo Metropolitan University, 6-6 Asahigaoka, Hino, Tokyo 191-0065, Japan. Tel.: +81 8031242452.
E-mail addresses: botzheim@tmu.ac.jp, botzheim@sze.hu (J. Botzheim).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2013.09.013

This technique effectively combines the evolutionary and the gradient based approaches eliminating their drawbacks by their combination, but it requires a large number of generations, so the time savings gained by the use of the parametric representation of fuzzy exponents instead of the extension principle can be significant as it will be shown in the experimental results section.

Several papers deal with different combinations of fuzzy and neural network techniques to better model the uncertain and imprecise problems [1,12,13]. The research on hybrid learning algorithms is also a popular topic in computational intelligence community [12–14]. Our motivation is, however, to accelerate the calculation of fuzzy exponent during the generations so that the total computation time of the training can be reduced.

2. Fuzzy neural networks

The topological structure of fuzzy neural network (FNN) is illustrated in Fig. 1. The network contains three layers. The input layer contains the fuzzy inputs $\tilde{x}_i$ with triangular shaped fuzzy membership functions in the form of (x_{iL}, x_{iC}, x_{iR}), where x_{iC} is the core of $\tilde{x}_i$ and $[x_{iL}, x_{iR}]$ represents its support interval. In the hidden layer m hidden neurons are applied. Between the input and hidden layers there are connection weights w_{ij} connecting the i-th input and j-th hidden neuron. There is also a bias input $x_0 = 1$ with weight w_{0j} to the j-th hidden neuron. There is an output layer with one crisp output y. Between the hidden and output layers there are weights v_j connecting the j-th hidden neuron with the output. The output of the fuzzy neural network can be computed as

$$y = \sum_{j=1}^{m} v_j \cdot \sigma\left(\sum_{i=0}^{n} \tilde{x}_i \cdot w_{ij}\right). \tag{1}$$

In Eq. (1) σ is the sigmoid function which in crisp case would be

$$\sigma(s) = \frac{1}{1+e^{-K\cdot s}}, \tag{2}$$

where K is the slope parameter of the sigmoid function. We introduce a novel calculation of fuzzy exponent in the sigmoid functions in a parametric form based on our former result [4]:

$$\begin{aligned} FE(a,\beta_L,\beta_C,\beta_R,\alpha,\lambda) = {} & \frac{1}{1+(1+\lambda)(1-\alpha)} a^{\beta_C} \\ & + \frac{1-\alpha}{1+(1+\lambda)(1-\alpha)} a^{\beta_C-(1-\alpha)(\beta_C-\beta_L)} \\ & + \frac{(1-\alpha)\lambda}{1+(1+\lambda)(1-\alpha)} a^{\beta_C+(1-\alpha)\lambda(\beta_R-\beta_C)}, \end{aligned} \tag{3}$$

where a is the base ($a=e$ in the sigmoid function), $\tilde{\beta} = (\beta_L, \beta_C, \beta_R)$ is the exponent, α is the level on which the fuzziness is considered, λ is the distortion parameter for suppressing the asymmetry caused by the right slope of the fuzzy exponent meaning that this

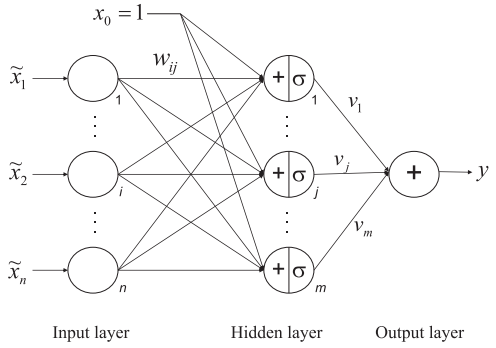

Fig. 1. Topological structure of fuzzy neural network.

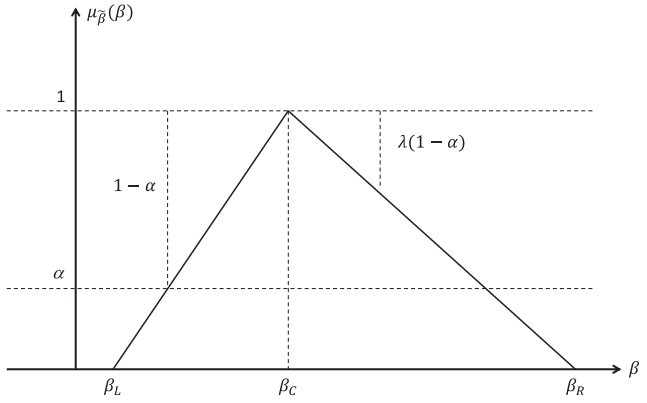

Fig. 2. Fuzzy exponent calculation.

part of the number plays a bigger role in the fuzzy power function than the left slope as illustrated in Fig. 2 [4].

For classical FNNs the learning is usually based on α-cuts [1], where on some α levels a traditional gradient based technique (e.g. the back-propagation algorithm) is used. In this case the exponent and the whole learning process have to be performed on each considered α level.

In our case the output of the sigmoid functions will be a crisp value because of the fuzzy exponent calculation. The used α is a parameter of the fuzzy exponent calculation, and this calculation has to be performed only on this specified α value. Comparing the complexity of the traditional way to our new approach we can state that although the single fuzzy exponent calculation needs a bit more cost than a crisp exponent calculation in the traditional approach we need to perform the calculation only once and not for all considered α levels.

The parameters of the architecture are the number of input neurons (n), the number of hidden neurons (m), the slope parameter (K), and the α and distortion (λ) parameters in the fuzzy exponent calculation.

3. Bacterial memetic algorithm

Nature inspired evolutionary optimization algorithms are often suitable for global optimization of even non-linear, high-dimensional, multi-modal, multi-objective, and discontinuous problems. Bacterial Evolutionary Algorithm (BEA) [15] is one of these techniques. BEA uses two operators, the bacterial mutation and the gene transfer operations. These new operators are based on the microbial evolution phenomenon. The bacterial mutation operation optimizes the chromosome of one bacterium, the gene transfer operation allows the transfer of information between the bacteria in the population. Each bacterium represents a solution for the original problem. BEA has been applied to a wide range of problems, for instance, optimizing fuzzy rule bases [15], feature selection [16], combinatorial optimization problems [17], automatic data clustering [18] and fuzzy rule selection [19].

Evolutionary algorithms are global searchers; however from practical point of view – concerning the running time – in most cases they can provide only a quasi-optimal solution to the problem, because their convergence speed is low. Local search approaches can give a more accurate solution; however they search for the solution only in the neighborhood of the previous candidates. Local search approaches might be useful in improving the performance of the basic evolutionary algorithm, which may find the global optimum with sufficient precision in this combined way. Combinations of evolutionary and local search methods are usually referred to as memetic algorithms [20].