Neural Computing and Applications (2021) 33:2705–2717
https://doi.org/10.1007/s00521-020-05133-w

ORIGINAL ARTICLE

Sparse regressions and particle swarm optimization in training high-order Takagi–Sugeno fuzzy systems

Krzysztof Wiktorowicz[1] · Tomasz Krzeszowski[1] · Krzysztof Przednowek[2]

Received: 13 November 2019 / Accepted: 16 June 2020 / Published online: 8 July 2020

Abstract
This paper proposes a method for training Takagi–Sugeno fuzzy systems using sparse regressions and particle swarm optimization. The fuzzy system is considered with Gaussian fuzzy sets in the antecedents and high-order polynomials in the consequents of the inference rules. The proposed method can be applied in two variants: without or with particle swarm optimization. In the first variant, ordinary least squares, ridge regression, or sparse regressions (forward selection, least angle regression, least absolute shrinkage and selection operator, and elastic net regression) determine the polynomials in the fuzzy system in which the fuzzy sets are known. In the second variant, we have a hybrid method in which particle swarm optimization determines the fuzzy sets, while ordinary least squares, ridge regression, or sparse regressions determine the polynomials. The first variant is simpler to implement but less accurate, the second variant is more complex, but gives better results. A new quality criterion is proposed in which the goal is to make the validation error and the model density as small as possible. Experiments showed that: (a) the use of sparse regression and/or particle swarm optimization can reduce the validation error and (b) the use of sparse regression may simplify the model by zeroing some of the coefficients.

Keywords Fuzzy systems · Least squares approximation · Sparse regression · Particle swarm optimization

1 Introduction

Fuzzy logic systems (or shortly, fuzzy systems), like other universal approximators, are capable of approximating any nonlinear function (mapping) with any degree of accuracy. Fuzzy systems offer a linguistic way of drawing conclusions because they are based on fuzzy IF-THEN rules. Calculation of the system outputs can be carried out in various ways. The literature mainly considers systems implemented with the use of Mamdani [22] and Takagi–Sugeno (T–S) [33] fuzzy inference. In a Mamdani system, the output of each rule is a fuzzy set, while in a T–S system, the output is a function of input variables. A T–S system is more computationally efficient since the defuzzification process is based on a weighted average rather than on a center of gravity.

Training fuzzy systems involve the selection of the number of rules (structure identification) and the determination of the parameters of the system. The first task is solved mainly by means of clustering methods [2, 26, 29], while the second one uses various optimization methods such as regressions [10, 35, 36], evolutionary algorithms [4, 6], particle swarm optimization [12, 15, 21, 26], and others [13, 43]. These methods are usually used in searching for the parameters for the antecedents and consequents of the fuzzy inference rules.

The main contributions of this paper are the use of sparse regressions in training high-order Takagi–Sugeno fuzzy systems and the proposition of a quality criterion that expresses a compromise between the accuracy of the model

The original version of this article was revised: An error in equation (39) in the pdf file of the article has been corrected.

✉ Krzysztof Wiktorowicz
kwiktor@prz.edu.pl

Tomasz Krzeszowski
tkrzeszo@prz.edu.pl

Krzysztof Przednowek
krzprz@ur.edu.pl

[1] Faculty of Electrical and Computer Engineering, Rzeszow University of Technology, al. Powstancow Warszawy 12, 35-959 Rzeszow, Poland

[2] College of Medical Sciences, Institute of Physical Culture Studies, University of Rzeszow, ul. Towarnickiego 3, 35-010 Rzeszow, Poland

and its sparsity. From the reviewed papers given in Sect. 2, it can be seen that currently, there are no applications of sparse regression methods for training fuzzy systems. In this article, we propose the use of sparse regressions that give sparse solutions, which means that some of the coefficients of the model are exactly zero. Such models are compact, and they are easier to interpret [27]. Moreover, sparse regressions provide regularization, which means that these methods can be used when the number of variables in the linear system exceeds the number of observations. In the proposed method, the premise parameters are determined manually or by a particle swarm optimization (PSO) algorithm, whereas the consequent parameters are defined by a sparse regression. The ordinary least squares (OLS) regression was used to build a fuzzy reference model. The proposed approach has been tested in three examples. This paper is a continuation of the work [38], where the sparsity of fuzzy models was not considered.

The structure of this article is as follows. Section 2 surveys the related work. Section 3 presents the proposed hybrid method for training Takagi–Sugeno fuzzy systems. This section contains the description of the high-order T–S system, training the consequent and antecedent parameters, the performance criteria, and the procedure for designing the fuzzy models. The experimental results and discussion are presented in Sect. 4. Finally, the conclusions are given in Sect. 5.

2 Related work

This chapter provides an overview of the literature on the use of PSO to train fuzzy systems. The review is divided into two parts: The first part discusses those papers in which PSO is used for training both the antecedents and consequents, whereas the second part discusses the papers where hybrid methods using PSO and regressions were applied.

2.1 Methods based on PSO

A two-phase swarm intelligence algorithm for zero-order Takagi–Sugeno–Kang (T–S–K) fuzzy systems was presented in [14]. The first phase aims to learn the structure of the fuzzy system and its parameters by ant colony optimization. This phase is used to find a good initial fuzzy rule base for further learning. In the second phase, the algorithm optimizes all of the free parameters in the fuzzy system using particle swarm optimization.

A multi-swarm cooperative particle swarm optimization method was proposed in [24], where the population consists of one master swarm and several slave swarms. The algorithm was used to automatically design the fuzzy identifier and the fuzzy controller in dynamical systems. Takagi–Sugeno fuzzy systems with Gaussian membership functions in the antecedents and linear functions in the consequents of the fuzzy rules were considered.

Khayat et al. [17] proposed a hybrid algorithm to design a fuzzy neural network to implement T–S fuzzy models. The algorithm consists of two phases. Firstly, the fuzzy model is generated by a coarse-tuning phase. In this phase, a fuzzy cluster validity index is used to determine the optimal number of clusters (rules), and fuzzy c-means is used to partition the input space. In the second phase, in order to adjust the parameters of the premise parts and consequent parts, genetic and PSO algorithms are used.

In [25], there is developed a method, called the knowledge acquisition with a swarm intelligence approach, for fuzzy rule evolution. They used a PSO algorithm to obtain the antecedents, consequents, and connectives (AND/OR) of the fuzzy rules. A Mamdani-type fuzzy rule-based system was used. The proposed method was tested on two examples: the control of an inverted pendulum, and scheduling in grid computing.

An algorithm that automatically extracts T–S fuzzy models from data using PSO was presented in [42]. The authors proposed a cooperative random learning PSO, where several subswarms search the space and exchange information. In their method, each fuzzy rule has a label that is used to decide whether the rule is included in the inference process or not. The antecedent parameters, the consequent parameters, and the rule labels are encoded in a particle.

In [5], there is proposed a heterogeneous multi-swarm PSO to identify the T–S fuzzy system. The T–S system proposed by the authors uses linear regression models in several subspaces to describe a nonlinear system. In the presented algorithm, the antecedent parameters and the consequent parameters of the T–S models are encoded in particles and are obtained simultaneously in the training process.

In [23], a hierarchical cluster-based multi-species PSO for building T–S–K fuzzy systems is presented. It is applied to a spatial analysis problem, in which the area under study is divided into several subzones. For each subzone, a zero-order or first-order fuzzy system is extracted from the dataset.

An immune coevolution PSO with multi-strategy for T–S fuzzy systems is proposed in [21]. The population consists of one elite subswarm and several normal subswarms. The parameters identified are the centers and widths of the membership functions (the antecedent parameters), and the coefficients of the local models (the consequent parameters).

In [26], there is presented a learning algorithm based on a hierarchical PSO (HPSO) to train the parameters of a T–S fuzzy model. First, an unsupervised fuzzy clustering algorithm is used to partition the data and identify the

antecedent parameters of the fuzzy system. Next, a self-adaptive HPSO algorithm is used to obtain the consequent parameters of the fuzzy system.

Jhang et al. [12] proposed a T–S–K-type fuzzy cerebellar model articulation controller (T-FCMAC) for solving various problems. To determine the parameters of the T-FCMAC, a group-based hybrid learning algorithm (GHLA) was used. The GHLA algorithm was developed by combining an improved quantum particle swarm optimization algorithm and the Nelder–Mead method. The authors also adopted the fuzzy C-mean clustering technique to improve the performance of quantum particle swarm optimization.

A new approach to optimize the Mamdani fuzzy systems without a need of any prior knowledge was proposed in [15]. The membership functions, the scaling factor parameters, and the fuzzy rule conclusions were optimized simultaneously with a mixed-coding PSO algorithm by combining a special monitoring function and a self-adaptive threshold.

2.2 Hybrid methods based on PSO and regressions

The approach proposed in [19] uses a hybrid learning method for T–S fuzzy systems. This method combines particle swarm optimization and recursive least squares (RLSE) to obtain a fuzzy approximation. The PSO is used to train the antecedent part of the T–S system, whereas the consequent part is trained by the RLSE method. The root-mean-square error is used as the cost function to measure the quality of the approximation.

An approach to building a type-2 neural-fuzzy system from a set of input–output data was proposed in [40]. First, a fuzzy clustering method is used to partition the dataset into clusters. Then, a type-2 fuzzy T–S–K rule is derived from each cluster. After the set of initial fuzzy rules is obtained, the parameters are refined using particle swarm optimization and a divide-and-merge-based least squares estimation.

In [41], there is proposed an approach to function approximation using robust fuzzy regression and particle swarm optimization. First, a fuzzy regression is used to construct a T–S–K fuzzy model. Next, particle swarm optimization is used to tune the parameters of that fuzzy model. As the fitness function, the root-mean-square error is used.

In [20], a self-learning complex neuro-fuzzy system that uses Gaussian complex fuzzy sets was presented. The knowledge base of the system consists of the T–S fuzzy rules with complex fuzzy sets in the antecedent part and linear models in the consequent part. The antecedent parameters and the consequent parameters are trained by the PSO algorithm and recursive least squares, respectively.

A method for fuzzy c-regression model clustering was presented in [28]. The proposed approach combines the advantages of two algorithms: clustering and particle swarm optimization. The fuzzy model used in this method is a T–S fuzzy system with local linear models. The consequent parameters of the fuzzy rules are estimated by the orthogonal least squares method.

In [39], a self-constructing radial basis function neural-fuzzy system was proposed, using particle swarm optimization for generating the antecedent parameters, and the least-Wilcoxon norm for the consequent parameters, instead of the traditional least squares estimation.

A T–S model based on particle swarm optimization and kernel ridge regression was presented in [2]. This algorithm works in two main steps. In the first step, the clustering based on the PSO algorithm separates the input data into clusters and obtains the antecedent parameters. In the second step, the consequent parameters are calculated using a kernel ridge regression. The proposed model was applied to generalized predictive control.

In [32], the adaptive chaos PSO algorithm (ACPSO) for identification of the T–S fuzzy model parameters using weighted recursive least squares was proposed. This approach is a compromise between the adaptive and chaos PSO algorithms. The ACPSO algorithm is used to optimize the parameters of the model; then, the obtained parameters are used to initialize the fuzzy c-regression model.

An identification method for the Takagi–Sugeno fuzzy model was proposed in [35]. Firstly, the fuzzy c-means algorithm is used to determine the optimal rule number. Next, the initial membership function and the consequent parameters are obtained by the PSO algorithm. To obtain the final parameters, a fuzzy c-regression model and orthogonal least square methods are applied.

In [36], there is proposed a complex fuzzy machine learning approach to function approximation. The PSO is used to optimize the premise parameters of the fuzzy model, while the recursive least squares estimator is used to find the consequents parameters.

3 Proposed method for training fuzzy systems

3.1 High-order Takagi–Sugeno fuzzy system

A Takagi–Sugeno (T–S) fuzzy system [33] with one input x and one output y is described by r fuzzy inference rules

$$R_j : \text{IF } x \in A_j(x) \quad \text{THEN } y = w_{mj}x^m + \cdots + w_{1j}x + w_{0j} \tag{1}$$

where $j = 1, 2, \ldots, r, A_j(x)$ is a fuzzy set, $m \geq 0$ is the degree of the polynomial, $w_{kj} \in \mathbb{R}$, and $k = 0, 1, \ldots, m$. The following definition given in [38] extends the concept of the T–S system in which zero or first-order polynomials are used.

Definition 1 The T–S system with the rules (1) is called:

- *zero-order* if $y = w_{0j}$, which means that the consequent functions are constants [33],
- *first-order* if $y = w_{1j}x + w_{0j}$, which means that the consequent functions are linear [33],
- *high-order* if $m \geq 2$, which means that the consequent functions are nonlinear [38].

In this paper, we use Gaussian membership functions for input fuzzy sets that can be unevenly spaced in the universe of discourse $x \in \mathbb{X} = [p_1, p_r]$ (see Fig. 1). These functions are given by

$$A_j(x) = \text{gauss}(x; p_j, \sigma_j) = \exp\left(-\frac{1}{2}\left(\frac{x - p_j}{\sigma_j}\right)^2\right) \tag{2}$$

where p_j is the peak of the function A_j, and $\sigma_j > 0$ is the width. The peaks p_j and the widths σ_j are written as the vectors $\mathbf{p} = [p_j] = [p_1, \ldots, p_r]$ and $\boldsymbol{\sigma} = [\sigma_j] = [\sigma_1, \ldots, \sigma_r]$, respectively.

The output of the T–S system is computed by

$$y = \frac{\sum_{j=1}^{r} A_j(x)\left(w_{mj}x^m + \cdots + w_{1j}x + w_{0j}\right)}{\sum_{j=1}^{r} A_j(x)} \tag{3}$$

Introducing the notion of a fuzzy basis function, formula (3) can be written in a compact form.

Definition 2 The *fuzzy basis function* (FBF) for the jth rule is the function $\xi_j(x)$ given by [37]

$$\xi_j(x) = \frac{A_j(x)}{\sum_{j=1}^{r} A_j(x)} \tag{4}$$

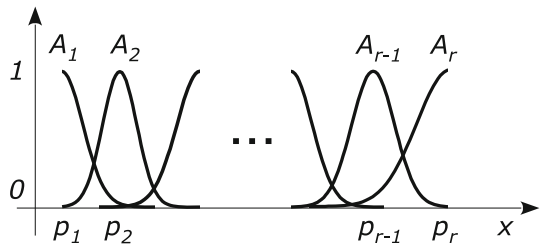

Fig. 1 Gaussian membership functions

Employing the definition in (4), the output of the T–S system can be written as

$$y = \sum_{j=1}^{r} w_{mj}\xi_j(x)x^m + \cdots + w_{1j}\xi_j(x)x + w_{0j}\xi_j(x) \tag{5}$$

In formula (5), the FBFs are multiplied by x^k; therefore, we define a modified fuzzy basis function.

Definition 3 The *modified FBF* (MFBF) for the jth rule is the function $h_{kj}(x)$ given by

$$h_{kj}(x) = \xi_j(x)x^k \tag{6}$$

where $k = 0, 1, \ldots, m$.

Applying (6), we obtain

$$y = \sum_{j=1}^{r} w_{mj}h_{mj}(x) + \cdots + w_{1j}h_{1j}(x) + w_{0j}h_{0j}(x) \tag{7}$$

We introduce the following vectors:

$$\mathbf{h}_j(x) = [h_{mj}(x), \ldots, h_{1j}(x), h_{0j}(x)] \tag{8}$$

$$\mathbf{w}_j = [w_{mj}, \ldots, w_{1j}, w_{0j}]^{\mathrm{T}} \tag{9}$$

where $\dim(\mathbf{h}_j) = \dim(\mathbf{w}_j) = m + 1$. The output of the T–S system (3) can now be written as

$$y = [\mathbf{h}_1(x), \ldots, \mathbf{h}_r(x)]\begin{bmatrix}\mathbf{w}_1 \\ \vdots \\ \mathbf{w}_r\end{bmatrix} = \mathbf{h}(x)\mathbf{w} \tag{10}$$

where

$$\mathbf{h}(x) = [\mathbf{h}_1(x), \ldots, \mathbf{h}_r(x)] \tag{11}$$

$$\mathbf{w} = [\mathbf{w}_1, \ldots, \mathbf{w}_r]^{\mathrm{T}} \tag{12}$$

The MFBFs are the elements of the vector $\mathbf{h}(x)$, and $\mathbf{w}$ is the vector of $p = r(m + 1)$ model parameters.

3.2 Training the consequent parameters

We assume as known the observations (x_i, y_i), where $i = 1, \ldots, n$. These observations are written as vectors $\mathbf{x} = [x_i]^{\mathrm{T}} = [x_1, \ldots, x_n]^{\mathrm{T}}$ and $\mathbf{y} = [y_i]^{\mathrm{T}} = [y_1, \ldots, y_n]^{\mathrm{T}}$. In order to apply regression methods, we introduce the regression matrix

$$\underset{n \times r(m+1)}{\mathbf{X}} = \begin{bmatrix}\mathbf{h}_1(x_1), \ldots, \mathbf{h}_r(x_1) \\ \mathbf{h}_1(x_2), \ldots, \mathbf{h}_r(x_2) \\ \vdots \\ \mathbf{h}_1(x_n), \ldots, \mathbf{h}_r(x_n)\end{bmatrix} \tag{13}$$

where $\mathbf{h}_j(x_i)$ is given by (8).

The matrix $\mathbf{X}$ will be used in determining consequent parameters of fuzzy rules by regressions such as ordinary least squares, ridge regression, or sparse regressions. The ordinary least squares, as the simplest method, will be applied further to build a fuzzy reference model.

3.2.1 Ordinary least squares

In the OLS, the cost function to be minimized is the sum of squared errors, given by

$$J_{\mathrm{OLS}} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - \mathbf{h}(x_i)\mathbf{w})^2 \tag{14}$$

where $\hat{y}_i = \mathbf{h}(x_i)\mathbf{w}$ is the estimated output of the system (see equation (10)) for the ith observation. The vector $\mathbf{w}$ contains the system parameters to be determined. The number of these parameters is equal to $p = \dim(\mathbf{w}) = r(m+1)$. The optimal solution is given by [1]

$$\mathbf{w} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y} \tag{15}$$

where $\mathbf{y} = [y_1, \ldots, y_n]^{\mathrm{T}}$. This is a batch least squares because the model parameters are computed directly from all the data contained in $\mathbf{X}$ and $\mathbf{y}$.

3.2.2 Ridge regression

In ridge regression [11], the cost function is the penalized sum of squared errors

$$J_{\mathrm{RIDGE}} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda\mathbf{w}^{\mathrm{T}}\mathbf{w} \tag{16}$$

$$= \sum_{i=1}^{n}(y_i - \mathbf{h}(x_i)\mathbf{w})^2 + \lambda\mathbf{w}^{\mathrm{T}}\mathbf{w} \tag{17}$$

where $\lambda \geq 0$ is a regularization parameter. The fuzzy model weights are given by

$$\mathbf{w} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y} \tag{18}$$

where $\mathbf{I}$ is the identity matrix. We apply the ridge regression because it can be used for ill-conditioned problems, that is when the matrix $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is close to singular. Ridge regression is a one-pass method, and therefore, it is very fast.

The OLS and ridge regressions have been implemented in MATLAB using a custom function.

3.2.3 Sparse regressions

The regressions described in this section are sparse, which means that some of the coefficients of the model are exactly zero [27]. These methods lead to reduced models that have a simpler structure and are easier to interpret. The following sparse methods are considered in this paper:

- Forward selection (FS)—This is a stepwise regression, i.e., variables are added one by one to the model. The algorithm starts with all coefficients equal to zero, and the next variable is chosen based on a certain criterion. For example, it can be the one with the highest correlation with the current residual vector [27].
- Least angle regression (LAR) [8, 27]—The LAR works similarly to the FS procedure, but instead of moving in the direction of one variable, the estimated parameters are calculated in a direction in which the angles with each of the variable currently in the model are equal. The LAR algorithm is the basis for other sparse methods, such as the least absolute shrinkage and selection operator and elastic net regression.
- Least absolute shrinkage and selection operator (LASSO) [27, 34]—This regression has a mechanism that implements a coefficient shrinkage and variable selection. The cost function combines the sum of the squared errors, and the penalty function is based on the L_1 norm:

$$J_{\mathrm{LASSO}}(\mathbf{w}, \delta) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \delta\|\mathbf{w}\|_1 \tag{19}$$

  where λ is a nonnegative regularization parameter.
- Elastic net (ENET) [27, 44]—The ENET regression combines the features of ridge regression and the LASSO. The cost function contains a penalty term related to both the L_1 and the L_2 norms:

$$J_{\mathrm{ENET}}(\mathbf{w}, \lambda, \delta) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2 + \delta\|\mathbf{w}\|_1 \tag{20}$$

  where λ and δ are nonnegative regularization parameters. To find the solution, the LARS-EN algorithm, which is based on the LARS algorithm [8], is used.

The sparse regressions have been implemented in MATLAB using the toolbox SpaSM [27].

3.2.4 Example of an application

To show the use of the ridge method when the problem is ill-defined (the number of observations is less than the number of predictors ($n < p$)), let us consider the following example of tuning the T–S system for a small amount of data. We have four observations ($n = 4$) in the form of vectors $\mathbf{x} = [1, 2, 3, 4]^{\mathrm{T}}$ and $\mathbf{y} = [6, 5, 7, 10]^{\mathrm{T}}$. For the input x, we define two fuzzy sets ($r = 2$) with the membership functions $A_1(x) = \mathrm{gauss}(x; 1, 1.274)$ and $A_2(x) = \mathrm{gauss}(x; 4, 1.274)$. In the consequent part of the rules (1), we assume the polynomials are linear, that is, $m = 1$. We can see that $n = p = 4$, which means the amount of data is equal to the number of regressors.

The regression matrix (13) is

$$\underset{4\times4}{\mathbf{X}} = \begin{bmatrix} \mathbf{h}_1(x_1), \mathbf{h}_2(x_1) \\ \mathbf{h}_1(x_2), \mathbf{h}_2(x_2) \\ \mathbf{h}_1(x_3), \mathbf{h}_2(x_3) \\ \mathbf{h}_1(x_4), \mathbf{h}_2(x_4) \end{bmatrix} = \begin{bmatrix} h_{11}(x_1), h_{01}(x_1), h_{12}(x_1), h_{02}(x_1) \\ h_{11}(x_2), h_{01}(x_2), h_{12}(x_2), h_{02}(x_2) \\ h_{11}(x_3), h_{01}(x_3), h_{12}(x_3), h_{02}(x_3) \\ h_{11}(x_4), h_{01}(x_4), h_{12}(x_4), h_{02}(x_4) \end{bmatrix} \tag{21}$$

For the data x_i, we obtain the following FBFs (4)

$$\begin{aligned} \xi_1(x_1) &= 0.9412 \quad & \xi_2(x_1) &= 0.0588 \\ \xi_1(x_2) &= 0.7159 \quad & \xi_2(x_2) &= 0.2841 \\ \xi_1(x_3) &= 0.2841 \quad & \xi_2(x_3) &= 0.7159 \\ \xi_1(x_4) &= 0.0588 \quad & \xi_2(x_4) &= 0.9412 \end{aligned} \tag{22}$$

Calculating the MFBFs (6) in the matrix (21), which is omitted here, we obtain

$$\underset{4\times4}{\mathbf{X}} = \begin{bmatrix} 0.9412, 0.9412, 0.0588, 0.0588 \\ 1.432, 0.7159, 0.5682, 0.2841 \\ 0.8523, 0.2841, 2.148, 0.7159 \\ 0.2353, 0.0588, 3.765, 0.9412 \end{bmatrix} \tag{23}$$

for which $\det(\mathbf{X}^{\mathrm{T}}\mathbf{X}) = 0.0523$. Using the OLS regression (15), the solution is

$$\mathbf{w} = [-1.897, 7.930, 1.715, 3.745]^{\mathrm{T}} \tag{24}$$

Assume now that we have only three observations, for example, $\mathbf{x} = [1, 3, 4]^{\mathrm{T}}$ and $\mathbf{y} = [6, 7, 10]^{\mathrm{T}}$. In this case, the regression matrix is given by

$$\underset{3\times4}{\mathbf{X}} = \begin{bmatrix} 0.9412, 0.9412, 0.0588, 0.0588 \\ 0.8523, 0.2841, 2.148, 0.7159 \\ 0.2353, 0.0588, 3.765, 0.9412 \end{bmatrix} \tag{25}$$

where $n<p$. For the matrix (25), we obtain $\det(\mathbf{X}^{\mathrm{T}}\mathbf{X}) = 1.144\mathrm{e}{-16}$, which means that $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is close to singularity and the fuzzy model can be unreliable. Applying the ridge regression (18) with $\lambda = 0.05$, we get $\det(\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}) = 0.2901$. The solution in this case is

$$\mathbf{w} = [1.040, 4.819, 2.575, -0.5194]^{\mathrm{T}} \tag{26}$$

Applying forward selection, we obtain three solutions in the coefficient path

$$\mathbf{w}_1 = [0, 0, 0, 0]^{\mathrm{T}} \tag{27}$$

$$\mathbf{w}_2 = [0, 0, 2.823, 0]^{\mathrm{T}} \tag{28}$$

$$\mathbf{w}_3 = [4.880, 0, 2.103, 0]^{\mathrm{T}} \tag{29}$$

Similarly, we can obtain solutions from the LAR, LASSO, and ENET regressions.

3.3 Training the antecedent parameters

A PSO algorithm is used to train the antecedent parameters. PSO was developed by Kennedy and Eberhart [7, 16]. It is based on the social behavior of living organisms that live in large groups. PSO has many advantages, among others, it can escape from local optima, it is easy to implement, and it has fewer parameters to adjust. PSO can be used for many optimization problems, such as in [9, 14, 18, 30, 41].

PSO consists of a set of solutions (particles), where each particle represents a point in a multi-dimensional space. A group of particles (a population) forms a swarm. Each particle behaves in a distributed manner, using its own intelligence and the group intelligence of the swarm. The particles move through the search space and exchange information to find the optimal solution. Each particle has a position (**x**) and velocity (**v**). In addition, each particle remembers its best position (**pbest**) and has access to the best position in the swarm (**gbest**). The learning process in the PSO method is based on two components:

- the cognition component: attracts particles toward its local most promising position.
- the social component: attracts particles toward the global best position discovered by the swarm.

The best solution is obtained by minimizing the objective function, calculated in this paper as the square root of the mean square error [19, 31].

The velocity $\mathbf{v}_k$ and position $\mathbf{x}_k$ of the kth particle are determined using the following equations [7]:

$$\mathbf{v}_k^{l+1} = \chi[\mathbf{v}_k^l + c_1\mathbf{r}_1(\mathbf{pbest}_k^l - \mathbf{x}_k^l) + c_2\mathbf{r}_2(\mathbf{gbest}^l - \mathbf{x}_k^l)] \tag{30}$$

$$\mathbf{x}_k^{l+1} = \mathbf{x}_k^l + \mathbf{v}_k^{l+1} \tag{31}$$

where χ is a constriction factor, here equal to 0.7298, $\mathbf{r}_1, \mathbf{r}_2$ are vectors of random numbers uniformly distributed within [0,1], l is the current iteration number, c_1 is the cognitive coefficient, and c_2 is the social coefficient ($c_1 = c_2 = 2.05$).

In this method, each of the particles represents the parameters of Gaussian fuzzy sets and contains the peaks $(p_2, \ldots, p_{r-1})$ and the widths of all membership $(\sigma 1, \ldots, \sigma_r)$ functions:

$$\underline{p_2 \mid p_3 \mid \ldots \mid p_{r-2} \mid p_{r-1} \mid \sigma_1 \mid \sigma_2 \mid \ldots \mid \sigma_{r-1} \mid \sigma_r} \tag{32}$$

We assume that the peaks p_1 and p_r are known; hence, the number of parameters to be chosen by the PSO algorithm is equal to $2r-2$.

To initialize the particles, we first distribute the peaks evenly in the interval (a, b), that is, the p_j is given by

$$p_j = a + (j-1)\delta \tag{33}$$

where $j = 1, 2, \ldots, r, \delta = (b-a)/(r-1), a = \min_i(x_i)$ and $b = \max_i(x_i)$. Next, we change $p_2, \ldots, p_{r-1}$ according to the formula

$$p_j := \max(\min(p_j + d_p \cdot \text{rand} - d_p/2, p_r), p_1) \tag{34}$$

where rand is a random number in the interval [0, 1], and d_p is the width of the initialization range for the peaks. The widths σ_j are initialized using the formula

$$\sigma_j := \max(\min(d_\sigma \cdot \text{rand}, \sigma_{\max}), \sigma_{\min}) \tag{35}$$

where $j \in 1, \ldots, r, d_\sigma$ is the width of the initialization range for σ_j and $0 < \sigma_{\min} < \sigma_{\max}$. During the optimization, we limit the peaks p_j to the interval $[p_1, p_r]$ and the widths σ_j of the Gaussian sets to the interval $[\sigma_{\min}, \sigma_{\max}]$.

3.4 Performance criteria

In this paper, we use a validation method in which the dataset is divided into two sets: a training set and a validation set. The training set is the set of observations used for learning, that is, for tuning the parameters (weights) of the fuzzy model. The validation set is also a set of observations, but it is disjoint from the training set. It is used to assess how well the model makes predictions based on new data. The performance criterion is calculated using the validation set, and it is calculated as the square root of the mean square error

$$\text{RMSE} = \sqrt{\frac{1}{V}\sum_{k=1}^{t}(y_k - \hat{y}_k)^2} \tag{36}$$

where V denotes the number of observations in the validation set, y_k denotes the kth target in the validation set, and $\hat{y}_k$ denotes the output of the fuzzy model obtained for kth input data x_k in the validation set.

The fuzzy models used in this paper may be *sparse*, which means they may have some coefficients equal to zero. The following definition introduces the notion of the sparsity of a fuzzy model.

Definition 4 The *sparsity* of a T–S fuzzy model is defined to be

$$z = \frac{nz}{r(m+1)} \tag{37}$$

where nz is the number of zero-valued coefficients in the polynomials, r is the number of rules, and m is the degree of the polynomial.

Using the definition of sparsity, we introduce the notion of density of a fuzzy model as follows.

Definition 5 The *density* of a T–S fuzzy model is defined as one minus the sparsity:

$$d = 1 - z \tag{38}$$

In this paper, we propose choosing the best T–S model by minimizing a quality criterion in which the goal is to make the validation error and the density as small as possible:

$$q = \alpha\frac{\text{RMSE}}{\text{RMSE}_{\text{OLS}}} + (1-\alpha)d \tag{39}$$

where $\alpha = 0.5$ and RMSE_{OLS} is the value of RMSE for the OLS regression that is treated as the reference method. The quality index (39) expresses a compromise between the accuracy of the model and its sparsity.

3.5 Procedure for designing fuzzy models

The proposed fuzzy models are designed in two variants: without and with particle swarm optimization. In the first variant, the regressions are used to determine the polynomials in a fuzzy system in which the fuzzy sets are known. In the second variant, the PSO determines the fuzzy sets, while the regressions determine the polynomials.

The following methods for building fuzzy models are employed in this paper:

- OLS: the method in which the fuzzy sets are defined by the user, while the polynomials are determined by the OLS regression.
- RIDGE: the method in which the fuzzy sets are defined by the user, while the polynomials are determined by the ridge regression.
- SR: the method in which the fuzzy sets are defined by the user, while the polynomials are determined by a sparse regression (SR), e.g., FS, LAR, LASSO, or ENET.
- PSO-OLS: the method in which the fuzzy sets are determined by the PSO algorithm, while the polynomials are determined by the OLS regression.
- PSO-RIDGE: the method in which the fuzzy sets are determined by the PSO algorithm, while the polynomials are determined by the ridge regression.

- PSO-SR: the method in which the fuzzy sets are determined by the PSO algorithm, while the polynomials are determined by a sparse regression.

The idea of the procedure for designing the fuzzy models is presented in Fig. 2.

In Block 1, we determine the Gaussian fuzzy sets. In the OLS, RIDGE, and SR methods, one proposition is generated in such a way that these sets are distributed evenly in the space $\mathbb{X}$, and the cross-point of two adjacent sets is equal to 0.5. In the PSO-OLS, PSO-RIDGE, and PSO-SR methods, ten propositions are generated by the PSO algorithm. The outputs of Block 1 are the vectors $\mathbf{p}$ and $\boldsymbol{\sigma}$.

In Block 2, we determine the regression matrix $\mathbf{X}$ (13).

In Block 3, we generate the coefficient path for one of the SR methods.

In Block 4, we validate the OLS, RIDGE, and PSO-RIDGE methods. As a result of validating the OLS method, we get the value for RMSE_{OLS}.

In Block 5, we validate the SR and PSO-SR methods. The validation is done along the coefficient path. For the PSO-SR method, we select from among ten fuzzy models the one with the smallest validation error RMSE. The quality index q is calculated in such a way that the smallest value is chosen, with the constraint that the RMSE is not greater than RMSE_{OLS}.

The results of the calculations in Blocks 4 and 5 are the optimal weights $(\mathbf{w}_{\text{opt}})$ of the fuzzy model.

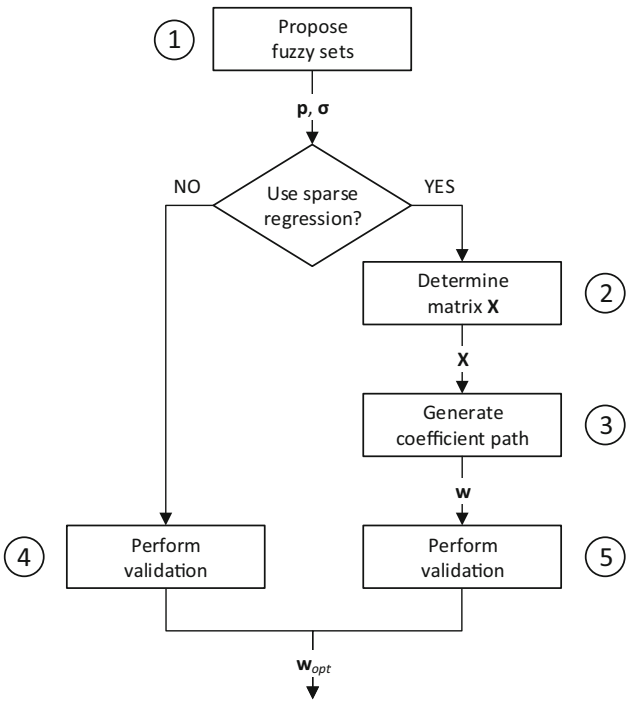

Fig. 2 The idea of the procedure for designing the fuzzy models

4 Experimental results and discussion

This section gives examples of the application of the developed methods to the approximation of nonlinear functions. The following parameters were adopted in all experiments: $\lambda = 1\text{e}{-}08$ for the ridge regression (18), $\lambda = 1\text{e}{-}10$ for the ENET regression (20), the number of iterations was 500, and the number of particles was 60 for the PSO algorithm. The parameter λ and the number of iterations were selected experimentally. The λ can prevent an ill-defined problem from occurring, but if it is too high, it can cause a deterioration of results. The number of particles was chosen based on [19]. We applied nine fuzzy rules, and the degree of the polynomial was two. The number of observations (x_i, y_i) was $n = 50$, and they were evenly distributed in the space $\mathbb{X}$. The dataset was divided into a training set of 40 observations and a validation set of 10 observations. The number of observations was chosen based on [3, 31]. The widths of the initialization range d_p, d_σ and the minimum and maximum values $\sigma_{\min}, \sigma_{\max}$ were chosen experimentally according to the range of input variables.

4.1 Experiment 1

We consider the nonlinear function [19]

$$y = 0.08(1.2(x-1))\cos(3x) + (x-(x-1)\cos(3x))\sin(x) \tag{40}$$

where $x \in [3, 7]$. The initialization parameters and limits were as follows: $d_p = 3.0, d_\sigma = 5.0, \sigma_{\min} = 0.1$, and $\sigma_{\max} = 5.0$. The experimental results are presented in Table 1. The smallest value of the quality index q, equal to 0.2955, was obtained for the PSO-ENET method. For this method, the validation error is smaller than the error for the reference model. Table 2 compares the parameters of the fuzzy systems obtained by the OLS method with those from the PSO-ENET method. It can be seen that the PSO-ENET method zeroed out 48% of the 27 coefficients. Based on this table, the fuzzy rules for the best model can be written as

$$\begin{aligned} R_1 &: \text{IF } x \in \text{gauss}(x, 3, 1.362) \\ &\quad \text{THEN } y = 1.085x^2 - 23.27x \\ R_2 &: \text{IF } x \in \text{gauss}(x, 3.939, 0.4366) \\ &\quad \text{THEN } y = 5.993x^2 - 31.31x \\ &\quad \ldots \\ R_9 &: \text{IF } x \in \text{gauss}(x, 7, 5) \\ &\quad \text{THEN } y = 0 \end{aligned} \tag{41}$$

It is seen that the fuzzy model PSO-ENET has zero polynomial in the consequent part of rule R_9. In Fig. 3, the approximation to the function is shown along with the approximation error for the PSO-ENET model.

Table 1 Performance comparison for Experiment 1

Algorithm	RMSE	z	q
OLS	1.635e−03	0	1
RIDGE	2.795e−02	0	9.045
FS	1.454e−03	0.0370	0.9260
LAR	1.514e−03	0.0370	0.9445
LASSO	1.298e−03	0.0370	0.8784
ENET	1.901e−03	0	1.081
PSO-OLS	*	*	*
PSO-RIDGE	1.762e−04	0	0.5032
PSO-FS	2.171e−04	0.5185	0.3071
PSO-LAR	2.363e−04	0.5185	0.3130
PSO-LASSO	1.690e−04	0.4444	0.3295
PSO-ENET	1.186e−04	0.4815	**0.2955**

The asterisk '*' means no solution

Table 2 Parameters of fuzzy systems in Experiment 1

Rule	p	σ	w_2	w_1	w_0
OLS					
R_1	3	0.2123	− 13.05	71.89	− 98.03
R_2	3.5	0.2123	20.82	− 148.3	262.5
R_3	4	0.2123	− 4.886	45.18	− 103.4
R_4	4.5	0.2123	− 30.30	260.5	− 561.4
R_5	5	0.2123	33.93	− 349.8	893.7
R_6	5.5	0.2123	18.79	− 189.9	469.8
R_7	6	0.2123	− 38.98	473.9	− 1441
R_8	6.5	0.2123	30.23	− 391.9	1272
R_9	7	0.2123	1.451	− 1.518	− 54.22
PSO-ENET					
R_1	3	1.362	1.085	− 23.27	0
R_2	3.939	0.4366	5.993	− 31.31	0
R_3	3.428	0.5482	− 4.775	0	0
R_4	4.397	0.4716	− 3.421	1.371	0
R_5	5.187	0.6116	− 2.290	− 32.52	0
R_6	3.496	1.999	14.70	− 4.811	0
R_7	6.047	0.6575	− 20.51	112.3	0
R_8	6.726	5	0	− 6.985	0
R_9	7	5	0	0	0

4.2 Experiment 2

In this experiment, we consider the nonlinear function [31]

$$y = \frac{(x-2)(2x-1)}{1+x^2} \tag{42}$$

where $x \in [-8, 12]$. The initialization parameters and limits were chosen to be $d_p = 12.0, d_\sigma = 8.0, \sigma_{\min} = 0.5$, and $\sigma_{\max} = 10.0$. The results of the applied methods are presented in Table 3. For the PSO-FS method, the quality index q equal to 0.4059 was the smallest value obtained for all simulations. The sparsity of this model was 26%. Table 4 presents the parameters of the fuzzy systems obtained using the OLS method and the PSO-FS method. The fuzzy rules for the best model can be written as

$$\begin{aligned} R_1 &: \text{IF } x \in \text{gauss}(x, -8, 0.9144) \\ &\quad \text{THEN } y = -0.0565x^2 - 0.7357x \\ R_2 &: \text{IF } x \in \text{gauss}(x, -3.040, 1.756) \\ &\quad \text{THEN } y = -0.2759x^2 - 2.456x - 1.725 \\ &\cdots \\ R_9 &: \text{IF } x \in \text{gauss}(x, 12, 3.921) \\ &\quad \text{THEN } y = 0.0117x^2 \end{aligned} \tag{43}$$

Figure 4 shows the approximation of the function and the error between the target function and the estimator for the PSO-FS model.

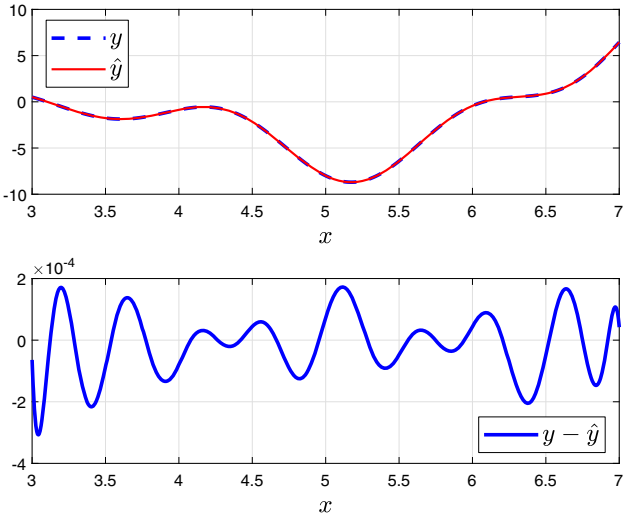

Fig. 3 Experiment 1: Approximation of the function and the error between the target function y and the estimator $\hat{y}$ for the PSO-ENET model in Table 2

4.3 Experiment 3

Here, the nonlinear function is given by [3]

$$y = 0.1 + 1.2x + 2.8\sin(4\pi x^2) \tag{44}$$

where $x \in [0, 1]$. The parameters for initialization and limits were as follows: $d_p = 0.3, d_\sigma = 1.0, \sigma_{\min} = 0.1$, and $\sigma_{\max} = 1.0$. The results of these experiments are presented in Table 5. The smallest value of q, equal to 0.2714, was obtained for the PSO-ENET algorithm. The sparsity of the chosen fuzzy model was $z = 52\%$. Table 6 presents the parameters of the fuzzy systems obtained using the OLS

Table 3 Performance comparison for Experiment 2

Algorithm	RMSE	z	q
OLS	4.144e−02	0	1
RIDGE	3.700e−02	0	0.9464
FS	3.526e−02	0.1482	0.8514
LAR	3.047e−02	0.0741	0.8306
LASSO	3.179e−02	0.1482	0.8095
ENET	3.179e−02	0.1482	0.8095
PSO-OLS	*	*	*
PSO-RIDGE	1.413e−03	0	0.5191
PSO-FS	2.941e−03	0.2593	**0.4059**
PSO-LAR	3.128e−03	0.2593	0.4081
PSO-LASSO	2.963e−03	0.2222	0.4246
PSO-ENET	3.004e−03	0.2593	0.4066

The asterisk '*' means no solution

Table 4 Parameters of fuzzy systems in Experiment 2

Rule	p	σ	w_2	w_1	w_0
OLS					
R_1	− 8	1.062	3.429	56.91	240.3
R_2	− 5.5	1.062	− 4.327	− 48.15	− 135.1
R_3	− 3	1.062	6.106	39.67	73.47
R_4	− 0.5	1.062	− 10.14	− 12.99	− 6.421
R_5	2	1.062	6.344	− 28.99	38.91
R_6	4.5	1.062	− 3.369	32.26	− 79.26
R_7	7	1.062	1.986	− 28.49	105.4
R_8	9.5	1.062	− 1.399	27.04	− 130.4
R_9	12	1.062	0.8924	− 22.08	138.6
PSO-FS					
R_1	− 8	0.9144	− 0.0565	− 0.7357	0
R_2	− 3.040	1.756	− 0.2759	− 2.456	− 1.725
R_3	− 1.624	0.5104	− 1.138	− 2.621	0
R_4	− 0.8196	0.5	− 3.031	− 3.186	1.676
R_5	− 1.919	0.8712	3.216	19.18	36.43
R_6	9.408	6.938	0.5113	2.321	− 1.786
R_7	8.237	4.424	0.1441	− 3.222	0
R_8	11.55	5.518	− 0.5326	0	0
R_9	12	3.921	0.0117	0	0

and PSO-ENET methods. The fuzzy rules for the PSO-ENET method are as follows:

$$
\begin{aligned}
&R_1: \text{IF } x \in \text{gauss}(x, 0, 0.2139)\\
&\quad \text{THEN } y = 14.51x - 2.028\\
&R_2: \text{IF } x \in \text{gauss}(x, 0.8145, 0.1)\\
&\quad \text{THEN } y = 126.3\\
&\quad \ldots\\
&R_9: \text{IF } x \in \text{gauss}(x, 1, 0.1)\\
&\quad \text{THEN } y = 32.53x^2 + 32.71
\end{aligned}
\tag{45}
$$

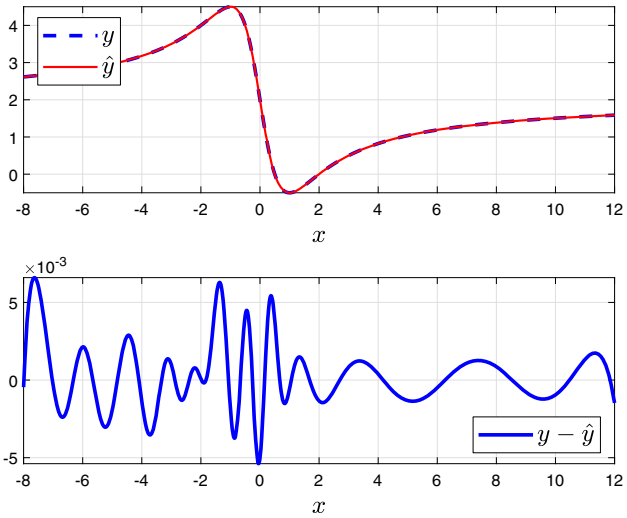

Fig. 4 Experiment 2: Approximation to the function and the error between the target function y and the estimator $\hat{y}$ for the PSO-FS model in Table 4

Table 5 Performance comparison for Experiment 3

Algorithm	RMSE	z	q
OLS	9.688e−03	0	1
RIDGE	4.039e−02	0	2.585
FS	8.539e−03	0.0741	0.9037
LAR	9.505e−03	0.0370	0.9720
LASSO	9.507e−03	0.0741	0.9537
ENET	9.945e−03	0	1.013
PSO-OLS	*	*	*
PSO-RIDGE	1.351e−04	0	0.5017
PSO-FS	1.935e−04	0.4074	0.3063
PSO-LAR	1.060e−03	0.5185	0.2954
PSO-LASSO	*	*	*
PSO-ENET	5.946e−03	0.5185	**0.2714**

The asterisk '*' means no solution

It is worth noting that the fuzzy model PSO-ENET has zero polynomial in the consequent part of rule R_6. The approximation of the function using the PSO-ENET method, together with the approximation error, is shown in Fig. 5.

4.4 Discussion

Based on the results presented in this section, it can be observed that:

- in the first variant (without PSO), the use of sparse regressions may reduce the validation error RMSE and

Table 6 Parameters of fuzzy systems in Experiment 3

Rule	p	σ	w_2	w_1	w_0
OLS					
R_1	0	0.0531	− 46.46	− 1.027	− 0.0060
R_2	0.125	0.0531	132.9	− 22.80	1.798
R_3	0.25	0.0531	− 97.88	64.29	− 7.759
R_4	0.375	0.0531	− 3.852	4.315	2.501
R_5	0.5	0.0531	− 425.3	374.3	− 81.12
R_6	0.625	0.0531	1030	− 1267	389.8
R_7	0.75	0.0531	− 922.5	1432	− 553.3
R_8	0.875	0.0531	121.9	− 291.9	162.9
R_9	1	0.0531	655.8	− 1249	595.1
PSO-ENET					
R_1	0	0.2139	0	14.51	− 2.028
R_2	0.8145	0.1	0	0	126.3
R_3	0.0090	0.9261	− 28.92	− 33.44	0
R_4	0.7227	0.1	− 81.64	0	0
R_5	0.8748	0.1	− 108.1	0	0
R_6	0.7960	0.1356	0	0	0
R_7	0.6655	0.1225	17.93	0	− 10.26
R_8	0.5463	0.2385	− 19.69	0	30.78
R_9	1	0.1	32.53	0	32.71

the quality index q compared to the fuzzy OLS-based reference model,

- in the second variant (with PSO), the experiments did not give results for the PSO-OLS method, but they were obtained for the PSO-RIDGE method,
- comparing the results in the second variant with the PSO-RIDGE method:
  - in the first experiment, a reduction in the error RMSE and the quality index q was obtained for the PSO-LASSO and PSO-ENET methods,
  - in the second and third experiments, the error RMSE for sparse regression methods was worse than for the PSO-RIDGE method, while the quality index q was better,
- the results of the error RMSE and the quality index q in the second variant are better than in the first variant,
- in each experiment, the model was simplified by reducing the number of polynomial coefficients: In the first experiment, a reduction of 48% was obtained, in the second—26%, while in the third 52%.

The obtained results show that the use of sparse regressions can reduce the validation error compared to the reference model and simplify fuzzy models by zeroing some coefficients. It should be emphasized that the developed method applies to T–S models [33] that have polynomials in their consequents, not fuzzy sets, as is the case of Mamdani models [22].

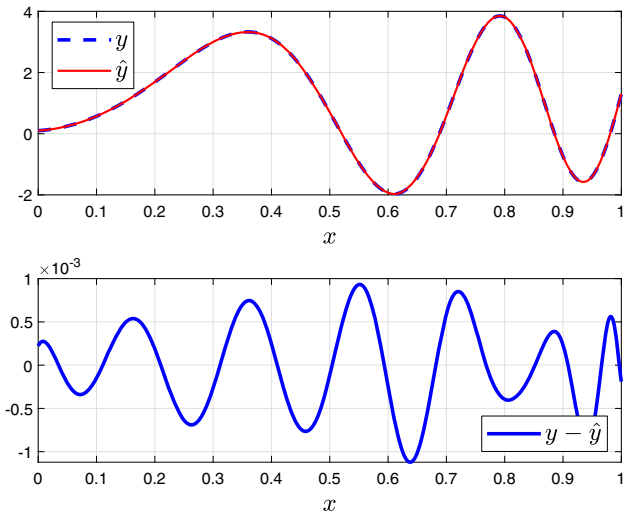

Fig. 5 Experiment 3: Approximation to the function and the error between the target function y and the estimator $\hat{y}$ for the PSO-ENET model in Table 6

5 Conclusions

Two methods of training high-order Takagi–Sugeno systems using sparse regressions and particle swarm optimization have been proposed. In the first, the antecedent parameters are set manually and in the second are set by a particle swarm optimization algorithm. In both variants, the consequent parameters are determined by a sparse regression. A fuzzy model based on the ordinary least squares regression is used as a reference method. To assess the quality of the fuzzy models, a quality criterion that expresses a compromise between the accuracy of the model and its sparsity was proposed. This criterion is based on the square root of the mean square error calculated for the validation set and the density of a fuzzy model. Compared with the reference method, the conducted experiments showed that: a) the use of sparse regressions and/or particle swarm optimization can reduce the validation error; b) the use of sparse regressions may simplify the fuzzy model by setting some of the coefficients to zero.

This paper uses a high-order T–S system to approximate the functions of one variable. Future work will focus on generalizing the proposed method for fuzzy systems with many input variables. Such systems can be applied to various applications, for example, for the approximation of functions with more than one variable, the identification of nonlinear models, the determination of inverse kinematics in robotics. Moreover, some work will be carried out toward the use of other optimization methods.

Compliance with ethical standards

Conflict of Interest The authors declare that they have no conflict of interest.

References

1. Bishop CM (2006) Pattern recognition and machine learning. Information science and statistics. Springer, New York
2. Boulkaibet I, Belarbi K, Bououden S, Marwala T, Chadli M (2017) A new T–S fuzzy model predictive control for nonlinear processes. Expert Syst Appl 88:132–151. https://doi.org/10.1016/j.eswa.2017.06.039
3. Bouzerdoum A (2000) Classification and function approximation using feed-forward shunting inhibitory artificial neural networks. In: Proceedings of the IEEE–INNS–ENNS international joint conference on neural networks (IJCNN 2000). Neural computing: new challenges and perspectives for the new millennium, vol 6, pp 613–618. https://doi.org/10.1109/IJCNN.2000.859463
4. Chen C, Liu Y (2018) Enhanced ant colony optimization with dynamic mutation and ad hoc initialization for improving the design of TSK-type fuzzy system. Comput Int Neurosci. https://doi.org/10.1155/2018/9485478
5. Cheung NJ, Ding XM, Shen HB (2014) Optifel: a convergent heterogeneous particle swarm optimization algorithm for Takagi–Sugeno fuzzy modeling. IEEE Trans Fuzzy Syst 22(4):919–933
6. Cortés-Antonio P, Batyrshin I, Martínez-Cruz A, Villa-Vargas LA, Ramírez-Salinas MA, Rudas I, Castillo O, Molina-Lozano H (2020) Learning rules for Sugeno ANFIS with parametric conjunction operations. Appl Soft Comput 89:106095. https://doi.org/10.1016/j.asoc.2020.106095
7. Eberhart RC, Shi Y (2000) Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of the 2000 congress on evolutionary computation (CEC 2000), vol 1, pp 84–88
8. Efron B, Hastie T, Johnstone I, Tibshirani R et al (2004) Least angle regression. Ann Stat 32(2):407–499
9. Fu Y, Ding M, Zhou C, Hu H (2013) Route planning for unmanned aerial vehicle (UAV) on the sea using hybrid differential evolution and quantum-behaved particle swarm optimization. IEEE Trans Syst Man Cybern Syst 43(6):1451–1465. https://doi.org/10.1109/TSMC.2013.2248146
10. Ge D, Zeng XJ (2019) A self-evolving fuzzy system which learns dynamic threshold parameter by itself. IEEE Trans Fuzzy Syst 27(8):1625–1637. https://doi.org/10.1109/TFUZZ.2018.2886154
11. Hoerl AE, Kennard RW (1970) Ridge regression: biased estimation for nonorthogonal problems. Technometrics 12(1):55–67
12. Jhang JY, Lin CJ, Li L (2019) Supervised and reinforcement group-based hybrid learning algorithms for TSK-type fuzzy cerebellar model articulation controller. Control Eng Appl Inform 21(2):11–21
13. Juang C, Hung C, Hsu C (2014) Rule-based cooperative continuous ant colony optimization to improve the accuracy of fuzzy system design. IEEE Trans Fuzzy Syst 22(4):723–735. https://doi.org/10.1109/TFUZZ.2013.2272480
14. Juang CF, Lo C (2008) Zero-order TSK-type fuzzy system learning using a two-phase swarm intelligence algorithm. Fuzzy Sets Syst 159(21):2910–2926
15. Kacimi MA, Guenounou O, Brikh L, Yahiaoui F, Hadid N (2020) New mixed-coding PSO algorithm for a self-adaptive and automatic learning of Mamdani fuzzy rules. Eng Appl Artif Intell 89:103417. https://doi.org/10.1016/j.engappai.2019.103417
16. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of the IEEE international conference on neural networks, vol 4. IEEE Press, Piscataway, pp 1942–1948
17. Khayat O, Ebadzadeh MM, Shahdoosti HR, Rajaei R, Khajehnasiri I (2009) A novel hybrid algorithm for creating self-organizing fuzzy neural networks. Neurocomputing 73(1):517–524. https://doi.org/10.1016/j.neucom.2009.06.013
18. Krzeszowski T, Przednowek K, Wiktorowicz K, Iskra J (2016) Estimation of hurdle clearance parameters using a monocular human motion tracking method. Comput Methods Biomech Biomed Eng 19(12):1319–1329 PMID: 26838547
19. Li C, Wu T (2011) Adaptive fuzzy approach to function approximation with PSO and RLSE. Expert Syst Appl 38(10):13266–13273
20. Li C, Wu T, Chan FT (2012) Self-learning complex neuro-fuzzy system with complex fuzzy sets and its application to adaptive image noise canceling. Neurocomputing 94:121–139
21. Lin G, Zhao K, Wan Q (2016) Takagi–Sugeno fuzzy model identification using coevolution particle swarm optimization with multi-strategy. Appl Intell 45(1):187–197
22. Mamdani EH, Assilian S (1975) An experiment in linguistic synthesis with a fuzzy logic controller. Int J Man–Mach Stud 7(1):1–13
23. Martino FD, Loia V, Sessa S (2014) Multi-species PSO and fuzzy systems of Takagi–Sugeno–Kang type. Inf Sci 267(Supplement C):240–251
24. Niu B, Zhu Y, He X, Shen H (2008) A multi-swarm optimizer based fuzzy modeling approach for dynamic systems processing. Neurocomputing 71(7–9):1436–1448
25. Prado RP, Garcia-Galan S, Exposito JEM, Yuste AJ (2010) Knowledge acquisition in fuzzy-rule-based systems with particle-swarm optimization. IEEE Trans Fuzzy Syst 18(6):1083–1097. https://doi.org/10.1109/TFUZZ.2010.2062525
26. Rastegar S, Araujo R, Mendes J (2017) Online identification of Takagi–Sugeno fuzzy models based on self-adaptive hierarchical particle swarm optimization algorithm. Appl Math Model 45(Supplement C):606–620
27. Sjöstrand K, Clemmensen L, Larsen R, Einarsson G, Ersbøll B (2018) SpaSM: a MATLAB toolbox for sparse statistical modeling. J Stat Softw 84(10):1–37. https://doi.org/10.18637/jss.v084.i10
28. Soltani M, Chaari A, Ben Hmida F (2012) A novel fuzzy C-regression model algorithm using a new error measure and particle swarm optimization. Int J Appl Math Comput Sci 22(3):617–628
29. Soltani M, Telmoudi AJ, Chaouech L, Ali M, Chaari A (2019) Design of a robust interval-valued type-2 fuzzy C-regression model for a nonlinear system with noise and outliers. Soft Comput 23(15):6125–6134. https://doi.org/10.1007/s00500-018-3265-z
30. Srinivasan D, Loo WH, Cheu RL (2003) Traffic incident detection using particle swarm optimization. In: Proceedings of the IEEE swarm intelligence symposium (SIS'03), pp 144–151

31. Sun TY, Tsai SJ, Tsai CH, Huo CL, Liu CC (2008) Nonlinear function approximation based on least Wilcoxon Takagi–Sugeno fuzzy model. In: 2008 Eighth international conference on intelligent systems design and applications, vol 1, pp 312–317
32. Taieb A, Soltani M, Chaari A (2018) A fuzzy C-regression model algorithm using a new PSO algorithm. Int J Adapt Control Signal Process 32(1):115–133. https://doi.org/10.1002/acs.2829
33. Takagi T, Sugeno M (1985) Fuzzy identification of systems and its applications to modeling and control. IEEE Trans Syst Man Cybern SMC–15(1):116–132
34. Tibshirani R (1996) Regression shrinkage and selection via the lasso. J R Stat Soc Ser B Methodol 58(1):267–288
35. Tsai SH, Chen YW (2018) A novel identification method for Takagi–Sugeno fuzzy model. Fuzzy Sets Syst 338:117–135
36. Tu CH, Li C (2018) Multiple function approximation—a new approach using complex fuzzy inference system. In: Nguyen NT, Hoang DH, Hong TP, Pham H, Trawiński B (eds) Intelligent information and database systems. Springer, Cham, pp 243–254
37. Wang L, Mendel JM (1992) Fuzzy basis functions, universal approximation, and orthogonal least-squares learning. IEEE Trans Neural Netw 3(5):807–814
38. Wiktorowicz K, Krzeszowski T (2020) Training high-order Takagi–Sugeno fuzzy systems using batch least squares and particle swarm optimization. Int J Fuzzy Syst 22(1):22–34. https://doi.org/10.1007/s40815-019-00747-2
39. Yang YK, Sun TY, Huo CL, Yu YH, Liu CC, Tsai CH (2013) A novel self-constructing radial basis function neural-fuzzy system. Appl Soft Comput 13(5):2390–2404
40. Yeh CY, Jeng WHR, Lee SJ (2011) Data-based system modeling using a type-2 fuzzy neural network with a hybrid learning algorithm. IEEE Trans Neural Netw 22(12):2296–2309
41. Ying KC, Lin SW, Lee ZJ, Lee IL (2011) A novel function approximation based on robust fuzzy regression algorithm model and particle swarm optimization. Appl Soft Comput 11(2):1820–1826 The Impact of Soft Computing for the Progress of Artificial Intelligence
42. Zhao L, Qian F, Yang Y, Zeng Y, Su H (2010) Automatically extracting T–S fuzzy models using cooperative random learning particle swarm optimization. Appl Soft Comput 10(3):938–944
43. Zhao W, Niu Q, Li K, Irwin GW (2013) A hybrid learning method for constructing compact rule-based fuzzy models. IEEE Trans Cybern 43(6):1807–1821
44. Zou H, Hastie T (2005) Regularization and variable selection via the elastic net. J R Stat Soc Ser B Stat Methodol 67(2):301–320