Approximation of two-variable functions using high-order Takagi–Sugeno fuzzy systems, sparse regressions, and metaheuristic optimization

Krzysztof Wiktorowicz¹ · Tomasz Krzeszowski¹

Published online: 5 September 2020
© The Author(s) 2020

Abstract

This paper proposes a new hybrid method for training high-order Takagi–Sugeno fuzzy systems using sparse regressions and metaheuristic optimization. The fuzzy system is considered with Gaussian fuzzy sets in the antecedents and high-order polynomials in the consequents of fuzzy rules. The fuzzy sets can be chosen manually or determined by a metaheuristic optimization method (particle swarm optimization, genetic algorithm or simulated annealing), while the polynomials are obtained using ordinary least squares, ridge regression or sparse regressions (forward selection, least angle regression, least absolute shrinkage and selection operator, and elastic net regression). A quality criterion is proposed that expresses a compromise between the prediction ability of the fuzzy model and its sparsity. The conducted experiments showed that: (a) the use of sparse regressions and/or metaheuristic optimization can reduce the validation error compared with the reference method, and (b) the use of sparse regressions may simplify the fuzzy model by zeroing some of the coefficients.

Keywords Fuzzy systems · Sparse regression · Metaheuristic optimization

1 Introduction

Many different methods have been developed for automatic training fuzzy systems from observed data. In this paper, we propose a novel approach to train Takagi–Sugeno fuzzy systems for function approximation. This approach is based on sparse regressions and metaheuristic optimization. Sparse regressions give sparse solutions, which means that some of the model coefficients are exactly zero. Such models are easier to interpret (Sjöstrand et al. 2018), more compact and therefore easier to implement. In addition, sparse regressions provide regularization, and therefore, they can be used when the problem is ill-conditioned (e.g., when the number of variables exceeds the number of observations). Metaheuristics

are modern nature-inspired algorithms widely used in global optimization problems (Glover and Kochenberger 2003). Metaheuristic means that in an algorithm, there is a “master strategy” at a higher level that guides heuristics applied in local search.

1.1 Related works

The literature on using metaheuristic optimization methods to train fuzzy systems is extensive. Further are discussed the papers in which hybrid methods using metaheuristics and regressions were applied. In such methods, the antecedents are trained by metaheuristic methods, while the consequents are trained by regressions.

One of the most commonly used algorithms to train fuzzy systems is particle swarm optimization (PSO). An approach presented in Li and Wu (2011) combines particle swarm optimization and recursive least squares estimator to obtain a fuzzy approximation. The PSO is used to train the antecedent part of the first-order T–S system, whereas the consequent part is trained by the RLSE method. Building a type-2 neural-fuzzy system was discussed in Yeh et al. (2011). In the first step, a fuzzy clustering method is used to partition the dataset into clusters. Then, a type-2 fuzzy Takagi–Sugeno–

Communicated by A. Di Nola.

✉ Krzysztof Wiktorowicz
kwiktor@prz.edu.pl

Tomasz Krzeszowski
tkrzeszo@prz.edu.pl

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

Kang (TSK) rule is derived from each cluster. The parameters are refined using PSO and a divide-and-merge-based least squares. In Ying et al. (2011), an approach to function approximation using robust fuzzy regression and particle swarm optimization is proposed. A fuzzy regression is used to construct the first-order TSK fuzzy model, whereas particle swarm optimization is used to tune its parameters. A self-learning complex neuro-fuzzy system that uses Gaussian complex fuzzy sets was proposed in Li et al. (2012). The knowledge base 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 particle swarm optimization algorithm and recursive least squares, respectively. In the paper (Soltani et al. 2012), a method for fuzzy c-regression model clustering was proposed. The method combines the advantages of two algorithms: clustering and particle swarm optimization. The consequent parameters of the first-order T-S fuzzy rules are estimated by the orthogonal least squares method. A self-constructing radial basis function neural-fuzzy system was proposed in Yang et al. (2013). The proposed method uses 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 two-step fuzzy model building algorithm based on particle swarm optimization and kernel ridge regression was presented in Boulkaibet et al. (2017). In the first step, the clustering based on particle swarm optimization 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. In Taieb et al. (2018), the adaptive chaos particle swarm optimization algorithm (ACPSO) using weighted recursive least squares was proposed. The ACPSO is used to optimize the parameters of the model, and then, the obtained parameters are used to initialize the fuzzy c-regression model. A fuzzy model identification method was proposed in Tsai and Chen (2018). Firstly, the fuzzy c-means algorithm is used to determine the rule number. Next, the initial fuzzy sets and the consequent parameters are obtained by particle swarm optimization. The final parameters are obtained using fuzzy c-regression and orthogonal least squares methods. In the paper (Tu and Li 2018), there is proposed a complex-fuzzy machine learning approach to function approximation. Particle swarm optimization is used to select the premise parameters of the first-order fuzzy model, while the recursive least squares estimator is used to find the consequent parameters.

Other commonly used methods are genetic algorithms (GA). A two-step approach to the construction of first-order fuzzy rules from data was proposed in Setnes and Roubos (2000). In the first step, fuzzy clustering and the weighted least squares are used to obtain an initial fuzzy model. In

the second step, this model is optimized by a real-coded genetic algorithm that allows simultaneous tuning of the rule antecedents and consequents. In Wang et al. (2005), a scheme based on multi-objective hierarchical GA (MOHGA) is proposed. This scheme is used to extract interpretable rule-based knowledge from data. First, fuzzy clustering is applied to generate an initial rule-based model. Then, the MOHGA and the recursive least squares estimator (RLSE) are used to obtain the optimized fuzzy models. A fuzzy modeling approach for the identification of nonlinear control processes was discussed in Yusof et al. (2011). This approach is based on a combination of genetic algorithm and recursive least squares. The antecedent parameters of the first-order T-S model are tuned by a genetic algorithm, whereas the consequent part is identified by recursive least squares estimator.

Various other approaches to train fuzzy models using metaheuristic optimization can be found in Almaraashi et al. (2016), Cordón et al. (2000, 2001), Cheung et al. (2014), Juang and Lo (2008), Khayat et al. (2009), Khosla et al. (2005, 2007), Lin (2008), Lin et al. (2016), Martino et al. (2014), Niu et al. (2008), Prado et al. (2010), Rastegar et al. (2017), Shihabudheen et al. (2018), Yanar and Akyürek (2011), Zhao et al. (2010). Advantages and disadvantages of the reviewed methods and the proposed method are presented in Table 1.

1.2 Contributions

From the literature review, it is seen that at most first-order polynomials are used in the consequent part to train fuzzy systems. In this paper, we propose to use high-order fuzzy systems for two-variable function approximation. In such systems, higher-order polynomials in the consequent part of rules are used, which can give greater flexibility in the selection of system parameters. Moreover, there is no use of sparse regressions for two-variable function approximation. Sparse regressions can generate sparse models (Sjöstrand et al. 2018), which are more compact, easier to interpret and implement. Summarizing, the main contributions of this paper can be stated as:

- The definition of high-order Takagi–Sugeno fuzzy systems with two input variables,
- The use of sparse regressions and metaheuristic optimization to train these systems.

In the proposed method, the premise parameters are determined manually or by metaheuristic optimization methods such as particle swarm optimization (PSO), genetic algorithm (GA), and simulated annealing (SA). The consequent parameters are calculated by ordinary least squares (OLS), ridge regression (RIDGE), and sparse regressions. The following sparse regressions have been used: forward selection

Table 1 Advantages and disadvantages of the methods used in related papers

Method	Advantages	Disadvantages
Manually chosen fuzzy sets with regression (Wiktorowicz and Krzeszowski 2020)	Very simple to use and implement Repeatability of obtained models Low computational cost	Generates less fitted models Regression may be ill-conditioned The number of fuzzy sets must be defined
Metaheuristics (Almaraashi et al. 2016; Cheung et al. 2014; Lin et al. 2016; Martino et al. 2014; Zhao et al. 2010)	Simple to use and implement	Large search space Difficulty in finding optimal parameters The number of fuzzy sets must be defined High computational cost Stochastic characteristic of the results
Metaheuristics with regression (Li and Wu 2011; Li et al. 2012; Taieb et al. 2018; Tu and Li 2018; Ying et al. 2011; Yusof et al. 2011; Yang et al. 2013)	The use of regression reduces the search space	Regression may be ill-conditioned The number of fuzzy sets must be defined High computational cost Stochastic characteristic of the results
Metaheuristics with clustering (Rastegar et al. 2017; Yanar and Akyürek 2011)	The use of clustering gives the initial structure of a system	Clustering complicates the algorithm High computational cost Stochastic characteristic of the results
Metaheuristics with clustering and regression (Boulkaibet et al. 2017; Soltani et al. 2012; Setnes and Roubos 2000; Tsai and Chen 2018; Wang et al. 2005; Yeh et al. 2011)	The use of clustering gives the initial structure of a system The use of regression reduces the search space	Clustering complicates the algorithm Regression may be ill-conditioned Stochastic characteristic of the results
Our approach	The use of sparse regression simplifies the model Using the ridge and sparse regressions prevents occurrence of ill-conditioned problem The use of regression reduces the search space The use of high-order system provides greater flexibility in system design	The number of fuzzy sets must be defined Stochastic characteristic of the results High computational cost Applying the high-order fuzzy system increases the number of parameters in the consequent part

(FS), least angle regression (LAR), least absolute shrinkage and selection operator (LASSO), and elastic net regression (ENET). The OLS regression was used as a reference model. This paper is a continuation of the work (Wiktorowicz and Krzeszowski 2020), where the approximation of one-variable functions was considered.

1.3 Paper structure

The structure of this paper is as follows. Section 2 describes the Takagi–Sugeno fuzzy system with two inputs and with high-order polynomials in the consequent parts of the fuzzy rules. Section 3 presents the training methods of the consequent parameters when using the OLS, RIDGE, and sparse regressions. Section 4 presents the training methods of the antecedent parameters when using the PSO, GA, and SA methods. The performance criterion is described in Sect. 5. Section 6 contains the design procedure for training fuzzy models. In Sect. 7, the experimental results are presented. Finally, the conclusions are given in Sect. 8.

2 High-order Takagi–Sugeno fuzzy system

We consider a Takagi–Sugeno (T–S) fuzzy system (Takagi and Sugeno 1985) with two inputs x_1, x_2 and one output y described by r fuzzy inference rules

$$\begin{aligned} R_j : & \text{IF } x_1 \in F_j(x_1) \text{ AND } x_2 \in G_j(x_2) \\ & \text{THEN } y = P_j(x_1, x_2), \end{aligned} \quad (1)$$

where $j = 1, 2, \dots, r$, $F_j(x_1)$, $G_j(x_2)$ are fuzzy sets, and $P_j(x_1, x_2)$ is the polynomial of degree d .

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

- *Zero-order* if $P_j(x_1, x_2) = b_j$, where $b_j \in \mathbb{R}$, which means that the consequent functions are constants (polynomial degree d is equal to zero) (Takagi and Sugeno 1985),
- *First-order* if $P_j(x_1, x_2) = w_{1j}x_1 + v_{1j}x_2 + b_j$, where $w_{1j}, v_{1j} \in \mathbb{R}$, which means that the consequent functions are linear (polynomial degree d is equal to one) (Takagi and Sugeno 1985),
- *High-order* if $P_j(x_1, x_2) = w_{mj}x_1^m + \dots + w_{1j}x_1 + v_{mj}x_2^m + \dots + v_{1j}x_2 + b_j$, where $m \geq 2$, $w_{kj}, v_{kj} \in \mathbb{R}$,

Fig. 1 Gaussian membership functions for the inputs x_1 and x_2 of the Takagi–Sugeno system. The parameters $p_1, \dots, p_{\rho}$ and $q_1, \dots, q_{\rho}$ are the points where the peaks of membership functions are placed

and $k = 2, 3, \dots, m$, which means that the consequent functions are nonlinear (polynomial degree d is greater than one).

In this paper, we use Gaussian membership functions that can be unevenly spaced in the universe of discourse (see Fig. 1). These functions are defined by

$$A_k(x_1) = \text{gauss}(x_1; p_k, \sigma_k) = \exp\left(-\frac{1}{2}\left(\frac{x_1 - p_k}{\sigma_k}\right)^2\right), \quad (2)$$

$$B_k(x_2) = \text{gauss}(x_2; q_k, \delta_k) = \exp\left(-\frac{1}{2}\left(\frac{x_2 - q_k}{\delta_k}\right)^2\right), \quad (3)$$

where $x_1 \in \mathbb{X}_1 = [p_1, p_{\rho}]$, $x_2 \in \mathbb{X}_2 = [q_1, q_{\rho}]$, $k = 1, 2, \dots, \rho$, ρ is the number of fuzzy sets for the inputs, p_k , q_k are the peaks, and σ_k , $\delta_k > 0$ are the widths. Using the definitions of fuzzy sets $A_k(x_1)$ and $B_k(x_2)$, the fuzzy rules (1) are written as table presented in Table 2, where $r = \rho^2$. The output of the T–S system is computed by

$$y = \frac{\sum_{j=1}^r F_j(x_1)G_j(x_2)P_j(x_1, x_2)}{\sum_{j=1}^r F_j(x_1)G_j(x_2)}. \quad (4)$$

Definition 2 Wang and Mendel (1992) The *fuzzy basis function* (FBF) for the j th rule is the function $\xi_j(x_1, x_2)$ given by

Table 2 Fuzzy rules for the Takagi–Sugeno system

j	$F_j(x_1)$	$G_j(x_2)$	$P_j(x_1, x_2)$
1	$A_1(x_1)$	$B_1(x_2)$	$P_1(x_1, x_2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$A_1(x_1)$	$B_{\rho}(x_2)$	
	$A_2(x_1)$	$B_1(x_2)$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$A_2(x_1)$	$B_{\rho}(x_2)$	
	$\vdots$	$\vdots$	
	$A_{\rho}(x_1)$	$B_1(x_2)$	
	$\vdots$	$\vdots$	
r	$A_{\rho}(x_1)$	$B_{\rho}(x_2)$	$P_r(x_1, x_2)$

$F_j(x_1)$ and $G_j(x_2)$ are the fuzzy sets for the inputs x_1 and x_2 , $P_j(x_1, x_2)$ is the high-order polynomial of x_1 and x_2 , r is the number of rules

$$\xi_j(x_1, x_2) = \frac{F_j(x_1)G_j(x_2)}{\sum_{j=1}^r F_j(x_1)G_j(x_2)}. \quad (5)$$

Applying (5), the output of the T–S system can be written:

– For the zero-order system as

$$y = \sum_{j=1}^r \xi_j(x_1, x_2)b_j, \quad (6)$$

– For the first-order and high-order systems as

$$y = \sum_{j=1}^r \xi_j(x_1, x_2)x_1^m w_{mj} + \dots + \xi_j(x_1, x_2)x_1 w_{1j} + \xi_j(x_1, x_2)x_2^m v_{mj} + \dots + \xi_j(x_1, x_2)x_2 v_{1j} + \xi_j(x_1, x_2)b_j. \quad (7)$$

Because in (7) the FBFs are multiplied by x_1^l and x_2^l where $l = 1, 2, \dots, m$, we define a modified fuzzy basis function.

Definition 3 The *modified FBF* (MFBF) for the j th rule is the function $h_{lj}(x_1, x_2)$ or $g_{lj}(x_1, x_2)$ given by

$$h_{lj}(x_1, x_2) = \xi_j(x_1, x_2)x_1^l, \quad (8)$$

$$g_{lj}(x_1, x_2) = \xi_j(x_1, x_2)x_2^l. \quad (9)$$

Applying (8) and (9) we obtain

$$y = \sum_{j=1}^r h_{mj}(x_1, x_2)w_{mj} + \dots + h_{1j}(x_1, x_2)w_{1j} + g_{mj}(x_1, x_2)v_{mj} + \dots + g_{1j}(x_1, x_2)v_{1j} + \xi_j(x_1, x_2)b_j. \quad (10)$$

We introduce the following vectors:

- For the zero-order system as

$$\mathbf{h}_j(x_1, x_2) = \xi_j(x_1, x_2), \quad (11)$$

$$\mathbf{w}_j = b_j, \quad (12)$$

- For the first-order and high-order systems as

$$\mathbf{h}_j(x_1, x_2) = [h_{mj}, \dots, h_{1j}, g_{mj}, \dots, g_{1j}, \xi_j], \quad (13)$$

$$\mathbf{w}_j = [w_{mj}, \dots, w_{1j}, v_{mj}, \dots, v_{1j}, b_j]^T, \quad (14)$$

where $\dim(\mathbf{h}_j) = \dim(\mathbf{w}_j^T) = 2d + 1$.

The output of the T-S system can now be written as

$$y = [\mathbf{h}_1(x_1, x_2), \dots, \mathbf{h}_r(x_1, x_2)] \begin{bmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_r \end{bmatrix} \quad (15)$$

$$= \mathbf{h}(x_1, x_2)\mathbf{w},$$

where

$$\mathbf{h}(x_1, x_2) = [\mathbf{h}_1(x_1, x_2), \dots, \mathbf{h}_r(x_1, x_2)], \quad (16)$$

$$\mathbf{w} = [\mathbf{w}_1, \dots, \mathbf{w}_r]^T. \quad (17)$$

The vector $\mathbf{w}$ contains $p = r(2d + 1)$ parameters of the T-S fuzzy model to be determined.

3 Training the consequent parameters

We assume as known the observations $[(x_1)_i, (x_2)_i]^T, y_i$, where $i = 1, \dots, n$ and n is the number of observations. We introduce the regression matrix

$$\mathbf{X}_{n \times r(2d+1)} = \begin{bmatrix} \mathbf{h}_1((x_1)_1, (x_2)_1), \dots, \mathbf{h}_r((x_1)_1, (x_2)_1) \\ \mathbf{h}_1((x_1)_2, (x_2)_2), \dots, \mathbf{h}_r((x_1)_2, (x_2)_2) \\ \vdots \\ \mathbf{h}_1((x_1)_n, (x_2)_n), \dots, \mathbf{h}_r((x_1)_n, (x_2)_n) \end{bmatrix}, \quad (18)$$

where $\mathbf{h}_j((x_1)_i, (x_2)_i)$ is given by (11) or (13).

3.1 Ordinary least squares

The cost function to be minimized in the OLS is the sum of squared errors

$$J_{\text{OLS}} = \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n (y_i - \mathbf{h}((x_1)_i, (x_2)_i)\mathbf{w})^2, \quad (19)$$

where $\hat{y}_i = \mathbf{h}((x_1)_i, (x_2)_i)\mathbf{w}$ is the estimated output of the system (see Eq. 15) for the i th observation. The optimal solution is given by Bishop (2006)

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}, \quad (20)$$

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

3.2 Ridge regression

The cost function in the ridge regression (Hoerl and Kennard 1970) is the penalized sum of squared errors

$$J_{\text{RIDGE}} = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \mathbf{w}^T \mathbf{w} \quad (21)$$

$$= \sum_{i=1}^n (y_i - \mathbf{h}((x_1)_i, (x_2)_i)\mathbf{w})^2 + \lambda \mathbf{w}^T \mathbf{w}, \quad (22)$$

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

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}, \quad (23)$$

where $\mathbf{I}$ is the identity matrix. The ridge regression is applied in this paper because it can be used for ill-conditioned problems, that is when the matrix $\mathbf{X}^T \mathbf{X}$ is close to singular. The ridge regression, similarly as the OLS, is a one-pass method, and therefore it is very fast.

3.3 Sparse regressions

The sparse regressions briefly described in this section allow the coefficients of a model to be exactly zero (Sjöstrand et al. 2018). These regressions lead to simplified models that are easier to interpret.

In the forward selection, that is an example of stepwise regression, the variables are added one by one to the model. In the beginning, all coefficients are equal to zero, and then a particular variable is chosen. The next variable to include can be chosen based on a number of criteria. For example, it can

be the one that has the highest correlation with the current residual vector (Sjöstrand et al. 2018).

The least angle regression (Efron et al. 2004; Sjöstrand et al. 2018) works similarly to the FS procedure, but the algorithm does not move in the direction of one variable. In the LAR, the estimated parameters are calculated in a direction in which the angles with each of the variables currently in the model are equal. This algorithm is the basis for other sparse methods, such as the LASSO and elastic net regression.

The least absolute shrinkage and selection operator regression (Sjöstrand et al. 2018; Tibshirani 1996) 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 based on the L_1 norm:

$$J_{\text{LASSO}}(\mathbf{w}, \lambda) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_1, \quad (24)$$

where λ is a nonnegative regularization parameter.

The elastic net regression (Sjöstrand et al. 2018; Zou and Hastie 2005) combines the features of the ridge regression and the LASSO. The cost function includes a penalty term related to both the L_1 and the L_2 norms:

$$J_{\text{ENET}}(\mathbf{w}, \delta, \lambda) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \delta \|\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_1, \quad (25)$$

where λ and δ are nonnegative regularization parameters. The solution is found by the LARS-EN algorithm, which is based on the LARS algorithm (Efron et al. 2004).

Example 1 Consider a simple regression problem for a small amount of data. We have four observations ($n = 4$) in the form of vectors $\mathbf{x} = [1, 2, 3, 4]^T$ and $\mathbf{y} = [6, 5, 7, 10]^T$. The goal is to build a regression model $y = ax + b$, where $\boldsymbol{\beta} = [a, b]$ is the vector of the model coefficients. To obtain a model with the intercept term (the constant b different from zero), we add the column of ones to the regression matrix, which has the form

$$\mathbf{X}_{4 \times 2} = \begin{bmatrix} 1, & 1 \\ 2, & 1 \\ 3, & 1 \\ 4, & 1 \end{bmatrix}. \quad (26)$$

It is easy to check that the OLS method gives the solution $y = 1.4x + 3.5$, where $a = 1.4$ and $b = 3.5$. Applying the FS, we obtain three solutions in the coefficient path

$$\boldsymbol{\beta}_1 = [0, 0], \quad \boldsymbol{\beta}_2 = [2.567, 0], \quad \boldsymbol{\beta}_3 = [1.4, 3.5]. \quad (27)$$

The LAR and the LASSO methods generate

$$\boldsymbol{\beta}_1 = [0, 0], \quad \boldsymbol{\beta}_2 = [2.45, 0], \quad \boldsymbol{\beta}_3 = [1.4, 3.5] \quad (28)$$

and using the ENET with $\delta = 0.1$ we obtain

$$\boldsymbol{\beta}_1 = [0, 0], \quad \boldsymbol{\beta}_2 = [2.682, 0], \quad \boldsymbol{\beta}_3 = [1.678, 3.421]. \quad (29)$$

We can see that in the solution $\boldsymbol{\beta}_2$, the coefficient b is exactly zero, that results from using the sparse regressions. The selection of one of the solutions is based on a specific criterion, e.g., cross-validation, Akaike's information criterion, or Bayesian information criterion (Sjöstrand et al. 2018).

4 Training the antecedent parameters

The following metaheuristic optimization methods were used to train the antecedent parameters: particle swarm optimization, (Eberhart and Shi 2000; Kennedy and Eberhart 1995; MathWorks 2019a), genetic algorithm (Holland 1992; Whitley 1994; MathWorks 2019a), and simulated annealing (Kirkpatrick et al. 1983; MathWorks 2019a).

4.1 Particle swarm optimization

Particle swarm optimization is a population-based algorithm developed by Kennedy and Eberhart and Shi (2000), Kennedy and Eberhart (1995). It is based on the social behavior of living organisms that live in large groups like birds flock or fish school. In PSO, a group of particles (a population) forms a swarm, in which each particle represents a hypothetical solution. The particle remembers its best position $\mathbf{pbest}$ and has access to the best position $\mathbf{gbest}$ in the swarm. The best local and global positions are selected using an objective function (Sect. 5). The learning scheme is based on two components:

- Cognition component—attracts particles toward the local best position,
- Social component—attracts particles toward the best position in the swarm.

The velocity $\mathbf{v}_k$ and the position $\mathbf{x}_k$ of the k th particle are calculated based on the following equations (Eberhart and Shi 2000; MathWorks 2019a):

$$\mathbf{v}_k^{l+1} = \omega \mathbf{v}_k^l + c_1 \mathbf{r}_1 (\mathbf{pbest}_k^l - \mathbf{x}_k^l) + c_2 \mathbf{r}_2 (\mathbf{gbest}^l - \mathbf{x}_k^l), \quad (30)$$

$$\mathbf{x}_k^{l+1} = \mathbf{x}_k^l + \mathbf{v}_k^{l+1}, \quad (31)$$

where ω is the inertia weight, $\mathbf{r}_1, \mathbf{r}_2$ are vectors of random numbers uniformly distributed within $[0, 1]$, l is the current iteration number, and c_1, c_2 are the cognitive and social coefficients, respectively.

4.2 Genetic algorithm

Genetic algorithm (Holland 1992; MathWorks 2019a; Whitley 1994) is a method for solving optimization problems inspired by the biological process of Darwinian evolution, where selection, crossover, and mutation play a major role. The GA repeatedly modifies a population to achieve new and possibly better solutions. In each generation of the GA, the individuals are randomly selected from the current population to be “parents” and used to obtain “children” for the next generation. In subsequent generations, the population “evolves” toward the optimal solution.

The GA uses three main types of rules to create the next generation from the current population:

- Selection—during this process, individuals called “parents” are selected through a fitness-based process. Individuals with a good value of the objective function (Sect. 5) are more often chosen for the next generation,
- Crossover (recombination)—combines two “parents” to form “children” for the next generation; it is analogous to the crossover that takes place during sexual reproduction in biology. The new individuals have the characteristics of both parents,
- Mutation—during the mutation process, an individual mutates that is random changes are introduced into the genotype. The purpose of this rule is to introduce diversity in the population that prevents the premature convergence of the algorithm.

Crossover and mutation characterize the explorative and exploitative features of GA. Maintaining a balance between these two features is crucial to speed up the search process and to achieve high-quality solutions.

4.3 Simulated annealing

Simulated annealing (Kirkpatrick et al. 1983; MathWorks 2019a) is a method for solving unconstrained and bound-constrained optimization problems. This method was originally inspired by the process of annealing in metallurgy. The SA models the process of heating a material and then gradually lowering the temperature in order to reduce defects. The goal is to move the system from the initial state to the state with minimum energy. As the algorithm runs, a new state is randomly generated and accepted with a certain probability. The acceptance probability is a function that depends on the energies of the two states and the temperature

$$p(\Delta E, T) = \frac{1}{1 + \exp(\Delta E/T)}, \quad (32)$$

where ΔE is the difference of energies of the present and previous solution ($\Delta E = E_{k+1} - E_k$) and T is the current temperature. The algorithm systematically decreases the temperature and stores the best state found so far. The energy determines how good the solution is, and it corresponds to the value of the objective function (Sect. 5).

5 Performance criterion

The objective function for all methods is the square root of the mean square error

$$\text{RMSE} = \sqrt{\frac{1}{V} \sum_{k=1}^V (y_k - \hat{y}_k)^2}, \quad (33)$$

where V denotes the number of observations in the validation set, y_k denotes the k th output data in the validation set, and $\hat{y}_k$ denotes the output of the fuzzy model obtained for the k th input data in the validation set. The fuzzy model used to calculate the estimate $\hat{y}_k$ is obtained based on the observations in the training set.

Fuzzy models used in this paper may be *sparse*, which means they may have some coefficients equal to zero. To describe the sparsity of a fuzzy model, we propose the following definition.

Definition 4 The *sparsity* of a T-S fuzzy model is defined as

$$S = \frac{z}{r(2d+1)}, \quad (34)$$

where $S \in [0, 1]$, z is the number of zero-valued coefficients in the polynomials, r is the number of rules, and d is the polynomial degree.

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

$$D = 1 - S. \quad (35)$$

In this paper, the best T-S model is chosen by minimizing a quality criterion in which the goal is to make the objective function (33) and the density as small as possible:

$$Q = \alpha \frac{\text{RMSE}}{\overline{\text{RMSE}}_{\text{OLS}}} + (1 - \alpha)D, \quad (36)$$

where $\alpha \in [0, 1]$. The $\overline{\text{RMSE}}_{\text{OLS}}$ is the mean value of RMSE for the OLS regression that is treated as the reference method. The quality index (36) expresses a compromise between the prediction ability of the model and its sparsity.

6 Design procedure for training fuzzy models

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

- Non-sparse methods:
 - 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,
 - 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,
 - GA-OLS: the method in which the fuzzy sets are determined by the GA, while the polynomials are determined by the OLS regression,
 - GA-RIDGE: the method in which the fuzzy sets are determined by the GA, while the polynomials are determined by the ridge regression,
 - SA-OLS: the method in which the fuzzy sets are determined by the SA algorithm, while the polynomials are determined by the OLS regression,
 - SA-RIDGE: the method in which the fuzzy sets are determined by the SA algorithm, while the polynomials are determined by the ridge regression,
- Sparse methods:
 - 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-SR: the method in which the fuzzy sets are determined by the PSO algorithm, while the polynomials are determined by a sparse regression,
 - GA-SR: the method in which the fuzzy sets are determined by the GA, while the polynomials are determined by a sparse regression,
 - SA-SR: the method in which the fuzzy sets are determined by the SA algorithm, while the polynomials are determined by a sparse regression.

The design procedure for training fuzzy models is presented in Fig. 2. In Block 1, the Gaussian fuzzy sets are proposed. In the OLS, RIDGE, and SR methods, one proposition is generated in such a way that these sets are distributed evenly in the spaces $\mathbb{X}_1$, $\mathbb{X}_2$, and the cross-point of two adja-

Fig. 2 The design procedure for training fuzzy models. In the first step, the fuzzy sets for the system inputs are selected, followed by the decision to use sparse regressions, and finally the models are validated. The results are the polynomial coefficients in the consequents of fuzzy rules

cent sets is equal to 0.5. In the PSO-OLS, PSO-RIDGE, GA-OLS, GA-RIDGE, SA-OLS, SA-RIDGE and PSO-SR, GA-SR, SA-SR methods, 10 propositions are generated by the PSO, GA or SA algorithms. The outputs of Block 1 are the vectors $\mathbf{p}$, σ , and $\mathbf{q}$, δ . In Block 2, the regression matrix $\mathbf{X}$ (18) is determined. In Block 3, the coefficient path for one of the SR methods is generated. In Block 4, no-sparse methods are validated. As a result of validating the OLS method, the value of RMSE_{OLS} in the quality criterion (36) is obtained. In Block 5, sparse methods are validated. The validation is done along the coefficient path. For all propositions, the RMSE, the sparsity S , and the quality index Q are calculated. Then, the smallest value of Q is chosen with the constraint that the RMSE is not greater than RMSE_{OLS} .

7 Experimental results

7.1 Experimental setup

This section gives examples of two-variable nonlinear function approximation. The following parameters were used in all experiments. The number of observations $[(x_1)_i, (x_2)_i]^T, y_i)$ was $n = 81$, and they were evenly distributed in the space $\mathbb{X}_1 \times \mathbb{X}_2$. The best method was selected using the Monte-

Carlo cross-validation (MCCV) (Picard and Cook 1984), in which the data set was divided randomly into some fraction of data to form the training set and to assign the rest of the points to the validation set. This process was repeated 10 times, generating new training and validation partitions in the proportion of 70% test data and 30% validation data. Statistical analysis was carried out using Wilcoxon signed rank test for differences in RMSE results between all methods and the reference method (OLS). For the inputs of the fuzzy system, three fuzzy sets were defined, which gave nine fuzzy inference rules. The widths of fuzzy sets were bounded in the intervals $[\sigma_{min}, \sigma_{max}] = [\delta_{min}, \delta_{max}] = [0.0849, 2.123]$. The degree of the polynomials in the consequent part was set to two. For the ridge regression (23), $\lambda = 1e-08$ and the ENET regression (25), $\delta = 1e-8$. The number of objective function evaluations was 6000. The parameter in the quality criterion (36) was $\alpha = 0.5$. For metaheuristic algorithms, default parameter values were adopted in accordance with the implementation contained in the Matlab toolbox. The experiments were carried out on a mobile computer equipped with Intel(R) Core(TM) i5-7200U and 8GB RAM.

7.2 Implementation

The function `regress` from the Matlab Statistics and Machine Learning Toolbox (MathWorks 2019b) has been used to apply the OLS regression. The ridge regression has been implemented in Matlab using a custom function.

The sparse regressions have been implemented in Matlab using the toolbox SpaSM (Sjöstrand et al. 2018). From this toolbox, the following functions have been used: `forwardselection`, `lar`, `lasso`, and `elasticnet`. These functions take the regression matrix $\mathbf{X}$ and the vector $\mathbf{y}$ as arguments. Moreover, the function `elasticnet` has the regularization parameter δ . As the output, the described functions return the solution path in the form of the coefficients $\mathbf{w}$, from which the best solution can be selected.

The metaheuristic methods have been implemented using the Global Optimization Toolbox in Matlab (MathWorks 2019a). From this toolbox, the following functions have been used:

`particleswarm`, `ga`, and `simulannealbnd`. These functions allow the solution to be obtained subject to the bounds defined by the user. They operate on the vec-

Table 3 Performance comparison for Experiment 1; $\overline{\text{RMSE}}$ is the mean of the validation error, std is the standard deviation, min is the minimum value, max is the maximum value, p is the p-value of Wilcoxon test, $\bar{S}$ is the mean of the model sparsity, $\bar{Q}$ is the mean of the quality index

Algorithm	$\overline{\text{RMSE}}$	std	min	max	p	$\bar{S}$	$\bar{Q}$
OLS	4.805e-02	4.563e-02	2.702e-02	17.70e-02	-	0	-
RIDGE	3.067e-02	3.771e-03	2.513e-02	3.884e-02	< 0.05	0	0.8192
FS	2.796e-02	4.611e-03	1.931e-02	3.247e-02	< 0.05	0.4422	0.5699
LAR	3.101e-02	6.465e-03	1.919e-02	4.063e-02	0.1056	0.4756	0.5849
LASSO	3.194e-02	5.447e-03	2.280e-02	4.063e-02	0.2324	0.5400	0.5624
ENET	3.195e-02	5.450e-03	2.280e-02	4.063e-02	0.2324	0.5400	0.5625
PSO-OLS	2.952e-05	4.093e-05	5.756e-06	11.69e-05	< 0.05	0	0.5003
PSO-RIDGE	2.952e-05	4.093e-05	5.756e-06	11.69e-05	< 0.05	0	0.5003
PSO-FS	3.539e-03	2.766e-03	8.732e-05	9.228e-03	< 0.05	0.7689	0.1524
PSO-LAR	3.700e-03	2.780e-03	6.908e-04	8.417e-03	< 0.05	0.7489	0.1641
PSO-LASSO	*	*	*	*	*	*	*
PSO-ENET	1.864e-03	1.237e-03	5.370e-04	4.643e-03	< 0.05	0.7489	0.1450
GA-OLS	3.101e-05	3.588e-05	7.599e-06	1.222e-04	< 0.05	0	0.5003
GA-RIDGE	3.101e-05	3.588e-05	7.599e-06	1.222e-04	< 0.05	0	0.5003
GA-FS	4.488e-03	2.629e-03	1.812e-03	9.691e-03	< 0.05	0.7644	0.1645
GA-LAR	2.954e-03	2.205e-03	7.822e-04	8.141e-03	< 0.05	0.7200	0.1707
GA-LASSO	*	*	*	*	*	*	*
GA-ENET	2.742e-03	1.458e-03	5.041e-04	5.452e-03	< 0.05	0.7489	0.1541
SA-OLS	3.380e-04	1.146e-04	1.534e-04	5.330e-04	< 0.05	0	0.5035
SA-RIDGE	3.380e-04	1.146e-04	1.534e-04	5.330e-04	< 0.05	0	0.5035
SA-FS	6.317e-03	3.436e-03	1.230e-03	1.180e-02	< 0.05	0.7711	0.1802
SA-LAR	4.993e-03	3.820e-03	1.115e-03	1.412e-02	< 0.05	0.7089	0.1975
SA-LASSO	*	*	*	*	*	*	*
SA-ENET	*	*	*	*	*	*	*

The asterisk '*' means no solution. The best result is marked in bold font

Table 4 Parameters of fuzzy systems in Experiment 1; p , q —peaks of membership functions, σ , δ —widths of membership functions, w_1 , w_2 , v_1 , v_2 , b —polynomial coefficients in the consequent part

Rule	p	σ	q	δ	w_2	w_1	v_2	v_1	b
OLS									
R_1	−1	0.4247	0	0.2123	5.296	11.51	−6.266	0.8213	6.488
R_2	−1	0.4247	0.5	0.2123	0.1319	−2.838	12.70	−10.61	0.5341
R_3	−1	0.4247	1	0.2123	24.57	56.02	−22.59	43.99	11.98
R_4	0	0.4247	0	0.2123	−5.941	0.0421	9.625	1.064	−0.5714
R_5	0	0.4247	0.5	0.2123	3.944	0.3770	−15.51	14.26	−3.450
R_6	0	0.4247	1	0.2123	−32.44	−0.2202	19.21	−39.19	16.17
R_7	1	0.4247	0	0.2123	5.328	−11.48	10.58	3.775	6.891
R_8	1	0.4247	0.5	0.2123	−1.701	6.309	−16.25	16.67	−8.504
R_9	1	0.4247	1	0.2123	25.98	−58.55	5.682	−15.92	45.31
PSO-ENET									
R_1	−0.9053	2.106	0.2191	0.4545	−1.991	0	0	0	0
R_2	−0.9053	2.106	0.5484	0.3625	5.924	0	0	0	0
R_3	−0.9053	2.106	0.9693	0.3919	−2.272	0	0	−0.0165	0
R_4	−0.1909	1.613	0.2191	0.4545	0	0	0	0	0
R_5	−0.1909	1.613	0.5484	0.3625	0	0	0	0	0
R_6	−0.1909	1.613	0.9693	0.3919	0	0	0	0	0
R_7	0.0284	1.959	0.2191	0.4545	−2.376	0.0035	0	0	−0.0147
R_8	0.0284	1.959	0.5484	0.3625	7.038	−0.0164	0	0	0.0394
R_9	0.0284	1.959	0.9693	0.3919	−2.719	0.0057	0	0	0

Fig. 3 Experiment 1: goal function graph

Fig. 4 Experiment 1: graph of the fuzzy estimator function obtained by the PSO-ENET method

tor that contains the parameters of Gaussian membership functions:

$$p_1 \mid \dots \mid p_\rho \mid q_1 \mid \dots \mid q_\rho \mid \sigma_1 \mid \dots \mid \sigma_\rho \mid \delta_1 \mid \dots \mid \delta_\rho$$

where $p_1, \dots, p_\rho, q_1, \dots, q_\rho$ are the peaks of membership functions, $\sigma_1, \dots, \sigma_\rho, \delta_1, \dots, \delta_\rho$ are the widths of membership functions.

7.3 Results of experiment 1

We consider the nonlinear function (Yeh et al. 2011)

$$y = x_1^2 \sin(\pi x_2), \quad (37)$$

where $x_1 \in [-1, 1]$ and $x_2 \in [0, 1]$. The results of Experiment 1 are presented in Table 3. The statistical analysis of the RMSE showed that most of the calculated models generate significantly different results ($p < 0.05$) compared to the

Fig. 5 Experiment 1: error between the goal function and the fuzzy estimator function obtained by the PSO-ENET method

OLS model. The exception is the LAR, LASSO and ENET models. The smallest value of the quality index $\bar{Q}$ is equal to 0.1450 and was obtained for the PSO-ENET method. For this method, the validation error $\overline{\text{RMSE}}$ is $1.864\text{e}-03$, and it is smaller than for the reference model for which this error is equal to $4.800\text{e}-02$. The sparsity $\bar{S}$ is 0.7489 that means that the PSO-ENET method zeroed out 75% of 45 coefficients. Thanks to this, the model is easier to interpret and implement. Table 4 contains the parameters of the fuzzy systems obtained by the OLS and the PSO-ENET methods. Based on this table, the fuzzy rules for the PSO-ENET model can be written as

$$\begin{aligned}
 R_1 : & \text{IF } x_1 \in \text{gauss}(x_1, -0.9053, 2.106) \\
 & \text{AND } x_2 \in \text{gauss}(x_2, 0.2191, 0.4545) \\
 & \text{THEN } y = -1.991x_1^2, \\
 R_2 : & \text{IF } x_1 \in \text{gauss}(x_1, -0.9053, 2.106) \\
 & \text{AND } x_2 \in \text{gauss}(x_2, 0.5484, 0.3625) \\
 & \text{THEN } y = 5.924x_1^2, \\
 & \dots \\
 R_9 : & \text{IF } x_1 \in \text{gauss}(x_1, 0.0284, 1.959) \\
 & \text{AND } x_2 \in \text{gauss}(x_2, 0.9693, 0.3919) \\
 & \text{THEN } y = -2.719x_1^2 + 0.0057x_1.
 \end{aligned} \tag{38}$$

Table 5 Training time comparison for Experiment 1 and Experiment 2

Algorithm	$\overline{\text{Time}} [s]$	Algorithm	$\overline{\text{Time}} [s]$
Experiment 1			
OLS	0.0086	GA-OLS	41.53
RIDGE	0.0090	GA-RIDGE	41.54
FS	0.0610	GA-FS	41.93
LAR	0.0732	GA-LAR	41.96
LASSO	0.1697	GA-LASSO	*
ENET	0.2422	GA-ENET	42.72
PSO-OLS	39.15	SA-OLS	52.13
PSO-RIDGE	39.15	SA-RIDGE	52.13
PSO-FS	39.61	SA-FS	52.55
PSO-LAR	39.61	SA-LAR	52.53
PSO-LASSO	*	SA-LASSO	*
PSO-ENET	40.28	SA-ENET	*
Experiment 2			
OLS	0.0095	GA-OLS	48.79
RIDGE	0.0134	GA-RIDGE	48.79
FS	0.0734	GA-FS	49.21
LAR	0.0702	GA-LAR	49.25
LASSO	*	GA-LASSO	50.57
ENET	0.2589	GA-ENET	50.48
PSO-OLS	47.99	SA-OLS	64.44
PSO-RIDGE	47.99	SA-RIDGE	64.44
PSO-FS	48.48	SA-FS	64.94
PSO-LAR	*	SA-LAR	64.90
PSO-LASSO	*	SA-LASSO	66.44
PSO-ENET	49.48	SA-ENET	65.87

The asterisk '*' means no solution

It is worth noting that in the PSO-ENET model, three rules (R_4 , R_5 , R_6) have the zero polynomial in the consequent part. Figures 3, 4, 5 show the goal function y , the estimator $\hat{y}$, and the approximation error $y - \hat{y}$ for the best model. The average time of training high-order T-S fuzzy systems using PSO-ENET method for one MCCV data subset was about 40.28 s (Table 5). The methods with manually chosen fuzzy sets (OLS, RIDGE, FS, LAR, LASSO, ENET) have the shortest calculation times, while the longest times have been obtained by algorithms using SA.

Table 6 Performance comparison for Experiment 2; $\overline{\text{RMSE}}$ is the mean of the validation error, std is the standard deviation, min is the minimum value, max is the maximum value, p is the p-value of Wilcoxon test, $\bar{S}$ is the mean of the model sparsity, $\bar{Q}$ is the mean of the quality index

Algorithm	$\overline{\text{RMSE}}$	std	min	max	p	$\bar{S}$	$\bar{Q}$
OLS	3.457e-01	2.550e-01	1.445e-01	9.588e-01	-	0	-
RIDGE	2.606e-01	1.485e-01	1.263e-01	5.187e-01	< 0.05	0	0.8769
FS	7.437e-02	1.386e-02	5.487e-02	9.488e-02	< 0.05	0.8756	0.1698
LAR	1.072e-01	9.910e-03	9.600e-02	1.290e-01	< 0.05	0.9356	0.1873
LASSO	*	*	*	*	*	*	*
ENET	1.082e-01	5.401e-03	9.801e-02	1.162e-01	< 0.05	0.9622	0.1754
PSO-OLS	3.687e-04	9.320e-05	2.672e-04	5.009e-04	< 0.05	0	0.5011
PSO-RIDGE	3.687e-04	9.320e-05	2.672e-04	5.009e-04	< 0.05	0	0.5011
PSO-FS	3.403e-02	1.920e-02	1.267e-02	7.570e-02	< 0.05	0.7956	0.1515
PSO-LAR	*	*	*	*	*	*	*
PSO-LASSO	*	*	*	*	*	*	*
PSO-ENET	1.723e-02	8.211e-03	4.127e-03	3.215e-02	< 0.05	0.7244	0.1627
GA-OLS	1.587e-04	9.430e-05	4.700e-05	3.013e-04	< 0.05	0	0.5003
GA-RIDGE	1.587e-04	9.430e-05	4.700e-05	3.013e-04	< 0.05	0	0.5003
GA-FS	3.726e-02	7.710e-03	2.865e-02	5.049e-02	< 0.05	0.8022	0.1528
GA-LAR	2.943e-02	1.524e-02	5.868e-03	4.783e-02	< 0.05	0.7356	0.1748
GA-LASSO	3.649e-02	1.768e-02	1.377e-02	5.949e-02	< 0.05	0.7489	0.1783
GA-ENET	2.860e-02	1.498e-02	1.377e-02	5.639e-02	< 0.05	0.7378	0.1725
SA-OLS	1.688e-03	5.591e-04	7.519e-04	2.510e-03	< 0.05	0	0.5034
SA-RIDGE	1.688e-03	5.591e-04	7.519e-04	2.510e-03	< 0.05	0	0.5034
SA-FS	3.121e-02	1.720e-02	6.798e-03	4.917e-02	< 0.05	0.7733	0.1585
SA-LAR	3.149e-02	1.334e-02	1.281e-02	5.217e-02	< 0.05	0.7489	0.1711
SA-LASSO	2.803e-02	1.506e-02	1.147e-02	5.557e-02	< 0.05	0.7133	0.1839
SA-ENET	2.182e-02	9.186e-03	7.292e-03	3.650e-02	< 0.05	0.7133	0.1749

The asterisk '*' means no solution. The best result is marked in bold font

7.4 Results of experiment 2

This experiment applies the nonlinear function (Yeh et al. 2011)

$$y = \sin(\pi x_1) \sin(\pi x_2), \quad (39)$$

where $x_1 \in [-1, 1]$ and $x_2 \in [0, 1]$. The results are presented in Table 6. The statistical analysis of the RMSE showed that all calculated models generate significantly different results ($p < 0.05$) compared to the OLS model. The smallest value of the quality index $\bar{Q}$ is equal to 0.1515 and was obtained for the PSO-FS method. For this method, the validation error $\overline{\text{RMSE}}$ is 3.403e-02, and the sparsity $\bar{S}$ is 0.7956. The PSO-FS method zeroed out 80% of 45 coefficients. The OLS method achieved the validation error $\overline{\text{RMSE}}$ equal to 3.457e-01. Table 7 contains the parameters of the fuzzy systems obtained by the OLS and the PSO-FS methods. The fuzzy rules for the PSO-FS model can be written as

$$\begin{aligned}
 R_1 : & \text{IF } x_1 \in \text{gauss}(x_1, -0.7318, 0.4725) \\
 & \text{AND } x_2 \in \text{gauss}(x_2, 0.6572, 1.025) \\
 & \text{THEN } y = 0, \\
 R_2 : & \text{IF } x_1 \in \text{gauss}(x_1, -0.7318, 0.4725) \\
 & \text{AND } x_2 \in \text{gauss}(x_2, 0.6618, 1.034) \\
 & \text{THEN } y = -7.586x_1, \\
 & \dots \\
 R_9 : & \text{IF } x_1 \in \text{gauss}(x_1, 1, 0.3468) \\
 & \text{AND } x_2 \in \text{gauss}(x_2, 0.5186, 0.3502) \\
 & \text{THEN } y = -7.655x_1^2 + 7.450.
 \end{aligned} \quad (40)$$

It is seen that two rules (R_1 and R_7) have the zero polynomial in the consequent part. Figures 6, 7 and 8 show the goal function y , the estimator $\hat{y}$, and the approximation error $y - \hat{y}$ for the best model. The average time of training high-order T-S fuzzy systems using the PSO-FS method for one MCCV data subset was about 48.48 s (Table 5). As in Experiment 1, the methods with manually chosen fuzzy sets (OLS, RIDGE, FS, LAR, LASSO, ENET) have the shortest calcu-

Table 7 Parameters of fuzzy systems in Experiment 2; p , q —peaks of membership functions, σ , δ —widths of membership functions, w_1 , w_2 , v_1 , v_2 , b —polynomial coefficients in the consequent part

Rule	p	σ	q	δ	w_2	w_1	v_2	v_1	b
OLS									
R_1	-1	0.4247	0	0.2123	4.794	11.64	53.93	3.130	8.401
R_2	-1	0.4247	0.5	0.2123	2.591	1.369	-57.28	57.33	-17.15
R_3	-1	0.4247	1	0.2123	26.88	59.32	45.42	-95.81	85.97
R_4	0	0.4247	0	0.2123	-7.663	-0.0414	28.90	1.408	-0.2997
R_5	0	0.4247	0.5	0.2123	0.3362	2.946	-31.22	30.64	-8.697
R_6	0	0.4247	1	0.2123	-26.27	3.442	30.60	-63.89	30.34
R_7	1	0.4247	0	0.2123	3.393	-9.364	-415.1	-6.422	0.6953
R_8	1	0.4247	0.5	0.2123	-0.5307	-1.685	288.6	-306.4	94.64
R_9	1	0.4247	1	0.2123	15.21	-35.50	-232.4	484.2	-234.7
PSO-FS									
R_1	-0.7318	0.4725	0.6572	1.025	0	0	0	0	0
R_2	-0.7318	0.4725	0.6618	1.034	0	-7.586	0	0	0
R_3	-0.7318	0.4725	0.5186	0.3502	0	0	0	0	-28.06
R_4	-0.9964	0.9916	0.6572	1.025	0	-4.925	0	0	0
R_5	-0.9964	0.9916	0.6618	1.034	-11.84	0	0	0	0.4208
R_6	-0.9964	0.9916	0.5186	0.3502	0	-10.55	0	0	13.37
R_7	1	0.3468	0.6572	1.025	0	0	0	0	0
R_8	1	0.3468	0.6618	1.034	2.035	0	0	0	0
R_9	1	0.3468	0.5186	0.3502	-7.655	0	0	0	7.450

Fig. 6 Experiment 2: goal function graph

Fig. 7 Experiment 2: graph of the fuzzy estimator function obtained by the PSO-FS method

lation times, while the longest times have been obtained by algorithms using SA.

8 Conclusions

A method of training high-order Takagi–Sugeno systems for two-variable function approximation has been proposed. The method is based on sparse regressions and metaheuristic optimization. The antecedent parameters of the fuzzy rules are

set manually or by metaheuristic optimization methods such as particle swarm optimization, genetic algorithm, or simulated annealing. The consequent parameters are determined by ordinary least squares, ridge regression or sparse regressions such as forward selection, least angle regression, least absolute shrinkage and selection operator or elastic net. Ordinary least squares regression is used as a reference method. A quality criterion based on sparsity measure has been proposed to assess the quality of the fuzzy models. Compared with the

Fig. 8 Experiment 2: error between the goal function and the fuzzy estimator function obtained by the PSO-FS method

reference method, the conducted experiments showed that: (a) the use of sparse regressions and/or metaheuristic optimization methods can reduce the validation error; (b) the use of sparse regressions may simplify the fuzzy model by setting some of the coefficients to zero.

Compliance with ethical standards

Conflict of interest The authors declare that there is no conflict of interests regarding the publication of this paper.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Almaraashi M, John R, Hopgood A, Ahmadi S (2016) Learning of interval and general type-2 fuzzy logic systems using simulated annealing: Theory and practice. *Inform Sci* 360:21–42. <https://doi.org/10.1016/j.ins.2016.03.047>
- Bishop CM (2006) Pattern recognition and machine learning. Information science and statistics. Springer, New York
- Boulkaibet I, Belarbi K, Bououden S, Marwala T, Chadli M (2017) A new T-S fuzzy model predictive control for nonlinear pro-

- cesses. *Expert Syst Appl* 88:132–151. <https://doi.org/10.1016/j.eswa.2017.06.039>
- 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. <https://doi.org/10.1109/TFUZZ.2013.2278972>
- Cordón O, Herrera F, Villar P (2000) Analysis and guidelines to obtain a good uniform fuzzy partition granularity for fuzzy rule-based systems using simulated annealing. *Int J Approx Reason* 25(3):187–215. [https://doi.org/10.1016/S0888-613X\(00\)00052-9](https://doi.org/10.1016/S0888-613X(00)00052-9)
- Cordón O, Herrera F, Villar P (2001) Generating the knowledge base of a fuzzy rule-based system by the genetic learning of the data base. *IEEE Trans Fuzzy Syst* 9(4):667–674. <https://doi.org/10.1109/91.940977>
- Eberhart RC, Shi Y (2000) Comparing inertia weights and constriction factors in particle swarm optimization. In: *Proceedings of the 2000 congress on evolutionary computation*, vol 1, pp 84–88. <https://doi.org/10.1109/CEC.2000.870279>
- Efron B, Hastie T, Johnstone I, Tibshirani R (2004) Others: least angle regression. *Ann Stat* 32(2):407–499
- Glover FW, Kochenberger GA (2003) Handbook of metaheuristics. Springer, Berlin
- Hoerl AE, Kennard RW (1970) Ridge regression: biased estimation for nonorthogonal problems. *Technometrics* 12(1):55–67
- Holland JH (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press, Cambridge
- Juang CFF, Lo C (2008) Zero-order TSK-type fuzzy system learning using a two-phase swarm intelligence algorithm. *Fuzzy Sets Syst* 159(21):2910–2926. <https://doi.org/10.1016/j.fss.2008.02.003>
- Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: *Proceedings of IEEE International Conference on Neural Networks*, vol 4. IEEE Press, Piscataway, NJ, pp 1942–1948
- Khayat O, Ebadzadeh MM, Shahdoosti HR, Rajaei R, Khajehnasiri I (2009) A novel hybrid algorithm for creating self-organizing fuzzy neural networks. *Neurocomputing* 73(1–3):517–524. <https://doi.org/10.1016/j.neucom.2009.06.013>
- Khosla A, Kumar S, Aggarwal KK (2005) A framework for identification of fuzzy models through particle swarm optimization algorithm. In: *2005 Annual IEEE India Conference-Indicon*, pp 388–391
- Khosla A, Kumar S, Ghosh KR (2007) A comparison of computational efforts between particle swarm optimization and genetic algorithm for identification of fuzzy models. In: *NAFIPS 2007—Annual meeting of the North American fuzzy information processing society*, pp 245–250. <https://doi.org/10.1109/NAFIPS.2007.383845>
- Kirkpatrick S, Gelatt CD Jr, Vecchi MP (1983) Optimization by simulated annealing. *Science* 220(4598):671–680. <https://doi.org/10.1126/science.220.4598.671>
- Li C, Wu T (2011) Adaptive fuzzy approach to function approximation with PSO and RLSE. *Expert Syst Appl* 38(10):13266–13273. <https://doi.org/10.1016/j.eswa.2011.04.145>
- Li C, Wu T, Chan FTT (2012) Self-learning complex neuro-fuzzy system with complex fuzzy sets and its application to adaptive image noise canceling. *Neurocomputing* 94:121–139. <https://doi.org/10.1016/j.neucom.2012.04.011>
- Lin CJ (2008) An efficient immune-based symbiotic particle swarm optimization learning algorithm for TSK-type neuro-fuzzy networks design. *Fuzzy Sets Syst* 159(21):2890–2909
- 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. <https://doi.org/10.1007/s10489-015-0752-0>
- Martino FD, Loia V, Sessa S, Di Martino F, Loia V, Sessa S (2014) Multi-species PSO and fuzzy systems of Takagi–Sugeno–Kang

- type. *Inform Sci* 267(Supplement C):240–251. <https://doi.org/10.1016/j.ins.2014.01.017>
- MathWorks (2019a) Global Optimization Toolbox: User's Guide
- MathWorks (2019b) Statistics and Machine Learning Toolbox: User's Guide
- 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. <https://doi.org/10.1016/j.neucom.2007.05.010>
- Picard RR, Cook RD (1984) Cross-validation of regression models. *J Am Stat Assoc* 79(387):575–583. <https://doi.org/10.1080/01621459.1984.10478083>
- Prado RP, García-Galán S, Muñoz Exposito JE, 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>
- 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 Modell* 45(Supplement C):606–620. <https://doi.org/10.1016/j.apm.2017.01.019>
- Setnes M, Roubos H (2000) GA-fuzzy modeling and classification: complexity and performance. *IEEE Trans Fuzzy Syst* 8(5):509–522. <https://doi.org/10.1109/91.873575>
- Shihabudheen KV, Mahesh M, Pillai GN (2018) Particle swarm optimization based extreme learning neuro-fuzzy system for regression and classification. *Expert Syst Appl* 92:474–484. <https://doi.org/10.1016/j.eswa.2017.09.037>
- Sjöstrand K, Clemmensen L, Larsen R, Einarsson G, Ersbøll B (2018) SpaSM: a MATLAB toolbox for sparse statistical modeling. *J Stat Softw Articles* 84(10):1–37. <https://doi.org/10.18637/jss.v084.i10>
- 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. <https://doi.org/10.2478/v10006-012-0047-0>
- 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>
- 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. <https://doi.org/10.1109/TSMC.1985.6313399>
- Tibshirani R (1996) Regression shrinkage and selection via the lasso. *J Royal Stat Soc Ser B (Methodological)* 58(1):267–288
- Tsai SHH, Chen YWW (2018) A novel identification method for Takagi–Sugeno fuzzy model. *Fuzzy Sets Syst* 338:117–135. <https://doi.org/10.1016/j.fss.2017.10.012>
- 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
- Wang H, Kwong S, Jin Y, Wei W, Man KF (2005) Multi-objective hierarchical genetic algorithm for interpretable fuzzy rule-based knowledge extraction. *Fuzzy Sets Syst* 149(1):149–186. <https://doi.org/10.1016/j.fss.2004.07.013>
- Wang L, Mendel JM (1992) Fuzzy basis functions, universal approximation, and orthogonal least-squares learning. *IEEE Trans Neural Netw* 3(5):807–814
- Whitley DCSU (1994) A genetic algorithm tutorial by Darrell Whitley. *Stat Comput* 4:65–85. <https://doi.org/10.1007/BF00175354>
- 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>
- Yanar TA, Akyürek Z (2011) Fuzzy model tuning using simulated annealing. *Expert Syst Appl* 38(7):8159–8169. <https://doi.org/10.1016/J.ESWA.2010.12.159>
- Yang YKK, Sun TYY, Huo CLL, Yu YHH, Liu CCC, Tsai CHH (2013) A novel self-constructing radial basis function neural-fuzzy system. *Appl Soft Comput* 13(5):2390–2404. <https://doi.org/10.1016/j.asoc.2013.01.023>
- 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. <https://doi.org/10.1109/TNN.2011.2170095>
- Ying KCC, Lin SWW, Lee ZJJ, Lee ILL (2011) A novel function approximation based on robust fuzzy regression algorithm model and particle swarm optimization. *Appl Soft Comput* 11(2):1820–1826. <https://doi.org/10.1016/j.asoc.2010.05.028>
- Yusof R, Abdul Rahman RZ, Khalid M, Ibrahim MF (2011) Optimization of fuzzy model using genetic algorithm for process control application. *J Franklin Inst* 348(7):1717–1737. <https://doi.org/10.1016/j.jfranklin.2010.10.004>
- 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. <https://doi.org/10.1016/j.asoc.2009.10.012>
- Zou H, Hastie T (2005) Regularization and variable selection via the elastic net. *J Royal Stat Soc Ser B (Stat Methodol)* 67(2):301–320

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.