Neural Computing and Applications (2022) 34:7473–7488
https://doi.org/10.1007/s00521-021-06843-5

ORIGINAL ARTICLE

Identification of time series models using sparse Takagi–Sugeno fuzzy systems with reduced structure

Krzysztof Wiktorowicz[1] · Tomasz Krzeszowski[1]

Received: 6 August 2021 / Accepted: 12 December 2021 / Published online: 5 January 2022

Abstract
Simplifying fuzzy models, including those for predicting time series, is an important issue in terms of their interpretation and implementation. This simplification can involve both the number of inference rules (i.e., structure) and the number of parameters. This paper proposes novel hybrid methods for time series prediction that utilize Takagi–Sugeno fuzzy systems with reduced structure. The fuzzy sets are obtained using a global optimization algorithm (particle swarm optimization, simulated annealing, genetic algorithm, or pattern search). The polynomials are determined by elastic net regression, which is a sparse regression. The simplification is based on reducing the number of polynomial parameters in the then-part by using sparse regression and removing unnecessary rules by using labels. A new quality criterion is proposed to express a compromise between the model accuracy and its simplification. The experimental results show that the proposed methods can improve a fuzzy model while simplifying its structure.

Keywords Time series prediction · Takagi–Sugeno fuzzy system · Elastic net regression · Global optimization

1 Introduction

In machine learning, model building is often based on the black-box principle, where the input and output observations are registered during the identification experiment. Based on these observations, the model is fitted to the data without paying particular attention to its mathematical structure. One possible solution in this area is the application of fuzzy models that use fuzzy rule-based inference. Various algorithms have been proposed for the automatic identification of fuzzy systems from observed data. We propose in this paper the use of sparse Takagi–Sugeno fuzzy systems with reduced structure and global optimization methods for model identification. Model simplification involves reducing the number of its parameters and reducing its structure by eliminating unnecessary rules. To reduce the number of parameters, we use a sparse regression that yields sparse solutions, in which some of the model coefficients are exactly equal to zero. Such models are more compact, easier to interpret [27], and thanks to this, easier to implement. Moreover, sparse regressions provide regularization, and therefore, they can be used in ill-conditioned problems (e.g., in the case when the number of variables exceeds the number of observations). To reduce the structure, we use the labels of rules that can change during the optimization process [34]. Global optimization is a branch of mathematics that provides methods for global solutions to problems that contain multiple maxima or minima. Some of these methods are modern metaheuristic algorithms widely used to solve global optimization problems [8]. Metaheuristic means that there is a 'master strategy' at a higher level of an algorithm that guides the heuristics applied in local search. In metaheuristic optimization algorithms, there is no guarantee that optimal solutions will be achieved. Usually, algorithms obtain near-optimal solutions, and this is the purpose of these. Moreover, many metaheuristics use some form of stochastic optimization, so that the found solution depends on the set of generated random variables.

✉ 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

From the literature review presented in the next section, it is seen that there is no use of sparse fuzzy systems with reduced structure for the identification of time series models. From the point of view of interpretation and implementation, simplifying fuzzy models, including models for predicting time series, is an important issue. Hence, the proposition of novel hybrid methods for time series prediction that utilize simplified Takagi–Sugeno fuzzy systems is the main motivation of our work. Our proposition concerns both simplifying the structure of the models (the number of fuzzy rules) and the number of parameters (in our case, the number of polynomial coefficients in the consequents of the fuzzy rules). Such a simplification, as proposed in this article, has not been considered elsewhere. Therefore, the main contributions can be formulated as follows:

- the proposition of hybrid methods that combine a sparse regression, rule labels, and a global optimization method to identify high-order Takagi–Sugeno fuzzy models with a reduced structure,
- the proposition of a new quality index that expresses a compromise between the accuracy of a model and its simplification,
- the use, for the first time, of sparse regression and rule labels for the time series prediction,
- the use of the proposed methods for the identification of time series models.

In the proposed methods, the if-part parameters are obtained by one of the global optimization methods, such as particle swarm optimization (PSO), simulated annealing (SA), genetic algorithm (GA), and pattern search (PS). The then-part parameters are determined by the ridge regression (RIDGE) and the sparse regression (SR) represented by the elastic net (ENET). The well-known adaptive neuro-fuzzy inference system (ANFIS) is used as a reference model.

The rest of this article is structured as follows. Section 2 surveys the related work. Section 3 describes a Takagi–Sugeno fuzzy system with high-order polynomials. Section 4 provides the training methods of the then-part parameters. Section 5 presents the training methods of the if-part parameters. The performance criterion and design procedure are described in Sect. 6. Section 7 presents issues related to the implementation of the discussed models. In Sect. 8, the experimental results are presented. At the end of the paper, the conclusions are presented in Sect. 9.

2 Related work

To identify time series models, global optimization methods, like particle swarm optimization [3, 4, 7, 16, 19, 20, 26, 28, 33, 34], genetic algorithms [3, 7, 9, 13, 16, 26, 28, 34], and simulated annealing [1, 2], are often used. The paper [19] presents a neuro-fuzzy system (NFS) with auto-regressive integrated moving average models and a novel hybrid learning method for resolving the problem of time series forecasting. The PSO algorithm and the recursive least squares estimator (RLSE) are combined in a hybrid manner to update the free parameters of the model. The PSO is used to update the antecedent parameters of the proposed predictor, and the RLSE is used to adjust the consequent parameters. Azad et al. [3] proposed an application of metaheuristic algorithms for training an artificial neural network (ANN) and ANFIS in order to predict the air temperature. To improve the performance of ANN and ANFIS, the PSO and GA were used. The best model turned out to be ANFIS-GA, which selected the most appropriate model inputs for forecasting the minimum, mean, and maximum air temperatures in different intervals. A new weighted fuzzy interpolative reasoning method was proposed in [4] for sparse fuzzy rule-based systems based on the slopes of fuzzy sets. To automatically learn the optimal weights of the antecedent variables of fuzzy rules, a PSO-based weights learning algorithm was utilized. The authors in [28] described an application of ensembles of interval type-2 fuzzy neural network (IT2FNN) models by utilizing hybrid learning algorithm techniques from NN models and fuzzy logic systems. For optimizing the parameter values in the membership functions of the fuzzy integrator, the PSO and GA were used. Lin et al. [20] proposed an IT2FNN based on a dynamic group cooperative PSO. The proposed model was tested in terms of the prediction accuracy and wall-following control of a mobile robot. In study [7], the optimization of type-2 fuzzy inference systems using GA and PSO was performed. The optimized fuzzy inference systems were used to estimate the type-2 fuzzy weights of backpropagation NNs. Khosravi et al. [16] developed a multilayer feed-forward NN, support vector regression, fuzzy inference system, ANFIS, group method of data handling (GMDH)-type NN, ANFIS optimized with PSO, and ANFIS optimized with GA to predict the time series wind speed data. The best results were obtained by the GMDH algorithm. A fuzzy NN model of the Takagi–Sugeno–Kang type was proposed in [13]. In this approach, the consequent part of fuzzy rules was a linear combination of input regressors and dominant wavelet neurons as a sub-jump wavelet NN. In order to obtain the wavelets for each sub-jump wavelet NN, the orthogonal least squares method

and GA were used. The authors in [26] presented an application of the most popular evolutionary algorithms (i.e., PSO, GA, artificial bee colony optimization, firefly algorithm, and whale optimization algorithm) to optimize the rule base of a fuzzy logic system. The authors in [1] presented a novel method for designing interval type-2 fuzzy logic systems in which the design parameters were tuned through the SA algorithm. In comparison with the conventional approach, this method has fewer parameters to be tuned, as only a single extra parameter is used to define the interval type-2 membership functions.

3 High-order Takagi–Sugeno fuzzy system

In our paper, we use a high-order Takagi–Sugeno (T–S) fuzzy system [32] with the inputs x_1, x_2 and output y, for which there are defined K fuzzy rules:

$$\begin{aligned} &\text{Rule } k: \text{If } x_1 \text{ is } G_k(x_1) \\ &\qquad \text{and } x_2 \text{ is } H_k(x_2) \\ &\qquad \text{then } y = W_k(x_1, x_2) \end{aligned} \tag{1}$$

where $G_k(x_1)$, $H_k(x_2)$ are the fuzzy sets for the inputs x_1, x_2, $W_k(x_1, x_2)$ is a two-variable polynomial of degree D, and $k = 1, 2, \ldots, K$. The polynomial $W_k(x_1, x_2)$ is of the form

$$\begin{aligned} W_k(x_1, x_2) = v_{Dk} x_1^D + \cdots + v_{1k} x_1 + w_{Dk} x_2^D + \cdots \\ + w_{1k} x_2 + c_k \end{aligned} \tag{2}$$

where $v_{dk}, w_{dk} \in \mathbb{R}$, $D \geq 1$, and $d = 1, 2, \ldots, D$.

The Gaussian membership functions for the inputs are defined as (Fig. 1)

$$A_z(x_1) = g(x_1; p_z, \sigma_z) = \exp\left(-\frac{1}{2}\left(\frac{x_1 - p_z}{\sigma_z}\right)^2\right) \tag{3}$$

$$B_z(x_2) = g(x_2; q_z, \delta_z) = \exp\left(-\frac{1}{2}\left(\frac{x_2 - q_z}{\delta_z}\right)^2\right) \tag{4}$$

where $x_1 \in [p_1, p_Z]$, $x_2 \in [q_1, q_Z]$, p_z, q_z are the peaks of the membership functions, $\sigma_z, \delta_z > 0$ are their widths, $z = 1, 2, \ldots, Z$, and Z is the number of fuzzy sets defined for the inputs. The fuzzy rules (1) are written as shown in Table 1, where $K = Z^2$. The rule labels l_k take the value 0 or 1 and are used to disable or enable the corresponding rule in fuzzy reasoning.

The output of the considered T–S system is calculated as [32]

$$y = \frac{\sum_{k=1}^{K} G_k(x_1) H_k(x_2) W_k(x_1, x_2)}{\sum_{k=1}^{K} G_k(x_1) H_k(x_2)} \tag{5}$$

$$= \sum_{k=1}^{K} \mu_k(x_1, x_2) W_k(x_1, x_2) \tag{6}$$

where

$$\mu_k(x_1, x_2) = \frac{G_k(x_1) H_k(x_2)}{\sum_{k=1}^{K} G_k(x_1) H_k(x_2)} \tag{7}$$

is the fuzzy basis function (FBF) [30]. In our application, the output of the T–S system is calculated as follows: the membership degrees $G_k(x_1)$ and $H_k(x_2)$ are multiplied, which gives the firing degree of a rule. These degrees are multiplied by the value of the polynomial $W_k(x_1, x_2)$ (this is a product implication). All partial results from the rules are aggregated with a weighted sum. To change the way of inference, we can change the product operation to, for example, the minimum operation (or, in general, to another t-norm).

Applying the FBF defined in (7), the system output can be written as

$$\begin{aligned} y = \sum_{k=1}^{K} \mu_k x_1^D v_{Dk} + \cdots + \mu_k x_1 v_{1k} + \mu_k x_2^D w_{Dk} + \cdots \\ + \mu_k x_2 w_{1k} + \mu_k c_k \end{aligned} \tag{8}$$

The FBFs in (8) are multiplied by x_1^d and x_2^d, so we introduce a modified fuzzy basis function (MFBF).

Definition 1 The MFBF [32] is the function $\beta_{dk}(x_1, x_2) = \mu_k(x_1, x_2) x_1^d$ or $\gamma_{dk}(x_1, x_2) = \mu_k(x_1, x_2) x_2^d$ defined for the kth rule.

Substituting β_{dk} and γ_{dk} for (8), we obtain

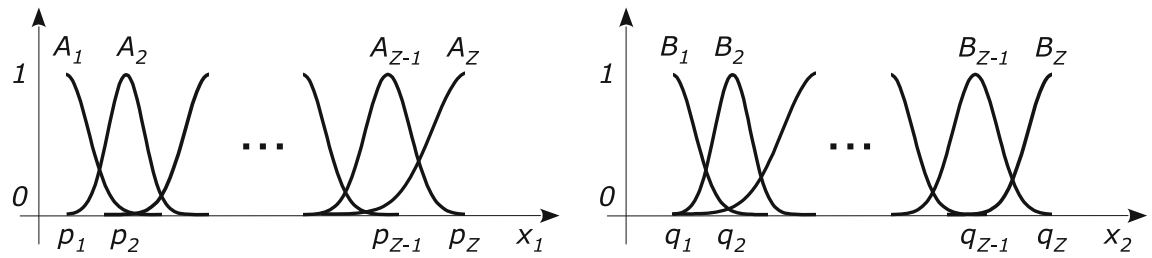

Fig. 1 Gaussian membership functions for inputs x_1 and x_2

Table 1 Fuzzy rules table

Rule	Label	$G_k(x_1)$	$H_k(x_2)$	$W_k(x_1, x_2)$
Rule 1	l_1	$A_1(x_1)$	$B_1(x_2)$	$W_1(x_1, x_2)$
		$\vdots$	$\vdots$	
		$A_1(x_1)$	$B_Z(x_2)$	
		$A_2(x_1)$	$B_1(x_2)$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
		$A_2(x_1)$	$B_Z(x_2)$	
		$\vdots$	$\vdots$	
		$A_Z(x_1)$	$B_1(x_2)$	
		$\vdots$	$\vdots$	
Rule K	l_K	$A_Z(x_1)$	$B_Z(x_2)$	$W_K(x_1, x_2)$

$$y = \sum_{k=1}^{K} \beta_{Dk} v_{Dk} + \cdots + \beta_{1k} v_{1k} + \gamma_{Dk} w_{Dk} + \cdots + \gamma_{1k} w_{1k} + \mu_k c_k \tag{9}$$

Introducing the vector

$$\boldsymbol{\xi}_k(x_1, x_2) = [\beta_{Dk}, \ldots, \beta_{1k}, \gamma_{Dk}, \ldots, \gamma_{1k}, \mu_k], \tag{10}$$

and

$$\mathbf{b}_k = [v_{Dk}, \ldots, v_{1k}, w_{Dk}, \ldots, w_{1k}, c_k]^T, \tag{11}$$

where $\dim(\boldsymbol{\xi}_k) = \dim(\mathbf{b}_k^T) = 2D + 1$, we can write the output of the T–S system as

$$y = [\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_K] \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_K \end{bmatrix} = \boldsymbol{\xi} \mathbf{b}^T \tag{12}$$

where $\boldsymbol{\xi} = [\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_K]$ and $\mathbf{b} = [\mathbf{b}_1^T, \ldots, \mathbf{b}_K^T]$. When calculating the output of the described fuzzy system, only rules with labels equal to 1 are taken into account.

4 Training the then-part of fuzzy rules

4.1 Regression matrix

Training the then-part of fuzzy rules involves calculating the coefficients of the polynomial (2) based on the regression matrix $\mathbf{X}$ defined in a similar way as in paper [32]. We consider data in the form of n observations $([(x_1)_i, (x_2)_i]^T, y_i)$, where $i = 1, \ldots, n$. We define the regression matrix

$$\underset{n \times K(2D+1)}{\mathbf{X}} = \begin{bmatrix} \boldsymbol{\xi}_1((x_1)_1, (x_2)_1), \ldots, \boldsymbol{\xi}_K((x_1)_1, (x_2)_1) \\ \boldsymbol{\xi}_1((x_1)_2, (x_2)_2), \ldots, \boldsymbol{\xi}_K((x_1)_2, (x_2)_2) \\ \vdots \\ \boldsymbol{\xi}_1((x_1)_n, (x_2)_n), \ldots, \boldsymbol{\xi}_K((x_1)_n, (x_2)_n) \end{bmatrix}, \tag{13}$$

where $\boldsymbol{\xi}_k((x_1)_i, (x_2)_i)$ is given by (10). The size of the regression matrix is $n \times K(2D + 1)$, where n is the number of training observations, K is the number of rules, and D is the polynomial degree.

4.2 Ridge regression

In the ridge regression, the loss function [10] is calculated from the equation

$$J_{\text{RIDGE}} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \mathbf{b}^T \mathbf{b} \tag{14}$$

where $\hat{y}_i$ is the estimated output of the system for the ith observation and $\lambda > 0$ is a regularization parameter. The vector of weights $\mathbf{b}$ of a fuzzy model is calculated as

$$\mathbf{b} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y} \tag{15}$$

where $\mathbf{y} = [y_1, \ldots, y_n]^T$ and $\mathbf{I}$ is the identity matrix. The vector $\mathbf{b}$ contains $K(2D + 1)$ then-part parameters of the T–S fuzzy model to be determined. It should be emphasized that in the ordinary least squares, the predictions and residuals are orthogonal. In the ridge regression, for $\lambda > 0$, this orthogonality is not guaranteed. But if we use this regression, we can deal with ill-defined problems (i.e., when the matrix $\mathbf{X}^T \mathbf{X}$ is close to singular), for example, in the presence of multicollinearity or when the number of predictors is greater than the number of observations.

4.3 Elastic net regression

The elastic net regression [35] is one of the sparse regressions that allow the coefficients of a model to be exactly zero [27]. Thus, it leads to simplified models that are easier to interpret. The elastic net integrates the ridge regression and LASSO [27, 29]. The lost function for this regression includes the penalty term relating to norms L_1 and L_2:

$$J_{\text{ENET}}(\mathbf{b}, \delta, \lambda) = \|\mathbf{y} - \mathbf{X}\mathbf{b}\|_2^2 + \delta \|\mathbf{b}\|_2^2 + \lambda \|\mathbf{b}\|_1 \tag{16}$$

where the parameters of regularization δ and λ are non-negative. The vector of solutions $\mathbf{b}$ is obtained by the LARS-EN method, which utilizes the LARS algorithm [6]. One use of elastic net regression is variable selection, but in our application, this regression is not used to select

variables, but to determine coefficients, some of which may be equal to zero, thus obtaining a sparse model.

5 Training the if-part of fuzzy rules

One of the following global optimization techniques is used to obtain the parameters of if-part of fuzzy rules: particle swarm optimization [5, 15, 24], simulated annealing [17, 24], genetic algorithm [11, 24, 31], and pattern search [12, 18].

5.1 Particle swarm optimization

Particle swarm optimization is a metaheuristic method introduced by Kennedy and Eberhart [5, 15] for stochastic search in a multidimensional space. The concept of the method was taken from the social behavior of animals that live in groups, like bird flocks, bee swarms, or fish schools. In the PSO, each individual in the population is called a particle and represents a potential solution. The way the particles move is modeled by the velocity $\mathbf{v}_k$ and position $\mathbf{x}_k$ [5, 24]:

$$\mathbf{v}_k^{i+1} = \omega \mathbf{v}_k^i + c_1 \mathbf{r}_1 \left(\mathbf{pb}_k^i - \mathbf{x}_k^i\right) + c_2 \mathbf{r}_2 \left(\mathbf{gb}^i - \mathbf{x}_k^i\right) \tag{17}$$

$$\mathbf{x}_k^{i+1} = \mathbf{x}_k^i + \mathbf{v}_k^{i+1} \tag{18}$$

where i is the current iteration number, k is the particle index, $\mathbf{r}_1$, $\mathbf{r}_2$ are vectors of uniformly distributed random numbers within [0,1], c_1, c_2 are, respectively, the cognitive and social coefficients, ω is the inertia weight, **pb** and **gb** are, respectively, the best local position of the kth particle and the best position in the swarm, which are selected using an objective function. After updating the particle position (18), the value of the objective function is calculated. Then, the determined value is compared with that of the objective function for **pb** and **gb**. If the new solution is better, then **pb** and **gb** are updated.

5.2 Simulated annealing

Simulated annealing [17, 24] is an algorithm used for solving optimization problems that are bound-constrained or unconstrained. The SA is inspired by the annealing process taking place in metallurgy. As the method runs, a new proposition of the state is obtained randomly and adopted with a certain probability according to the function

$$\text{prob}(\Delta E, T) = \frac{1}{1 + \exp(\Delta E / T)} \tag{19}$$

where T is the current temperature, and $\Delta E = E_{k+1} - E_k$ is the difference in the energies between the present and previous solutions. The energy describes how good is the proposed solution and corresponds to the value of the objective function. In the optimization process, the SA systematically decreases the temperature from an initial positive value to zero and remembers the best state found so far.

5.3 Genetic algorithm

Genetic algorithm [11, 24, 31] is a well-known optimization method inspired by the process of evolution. In the GA, a population is repeatedly modified to obtain new and better solutions. In each generation of the algorithm, the individuals in the population are randomly chosen to be 'parents' and used to generate 'children' for the next stage. In the process of optimization, the population 'evolves' in order to find the optimal solution. To create the next generation from the current population, the GA uses three main types of rules: selection, crossover (recombination), and mutation. During the selection, individuals called 'parents' are chosen using a fitness-based process. The crossover integrates two 'parents' to generate 'children' for the next generation. In the mutation phase, an individual mutates, which means that in the genotype random changes are introduced.

5.4 Pattern search

Pattern search (also known as direct search) [12, 18] is a method that can be applied to both constrained and unconstrained optimization problems. The method does not require a gradient, which means that it is a derivative-free algorithm. The PS searches for a function minimum based on an adaptive pattern. At each iteration, the algorithm tests multiple points that are placed near the current point, and moves the pattern to the point that best minimizes its objective function. The direction of this move is chosen according to a specified poll algorithm. The pattern shrinks in size if none of the proposed points is better than the current point. The method can run until the desired accuracy has been achieved or the algorithm reaches a maximum number of iterations.

6 Performance criterion and design procedure

6.1 Performance criterion

In this paper, the accuracy of a fuzzy model is described by two indices: the training error RMSE_t, obtained for the training data, and the validation error RMSE_v, obtained for the validation data of the form

$$\mathrm{RMSE}_t = \sqrt{\frac{1}{T}\sum_{k=1}^{T}(y_k - \hat{y}_k)^2} \tag{20}$$

$$\mathrm{RMSE}_v = \sqrt{\frac{1}{V}\sum_{k=1}^{V}(y_k - \hat{y}_k)^2} \tag{21}$$

where T is the number of observations in the training set, V is the number of observations in the validation set, y_k is the output data, and $\hat{y}_k$ is the predicted output.

The T–S fuzzy model is simplified in two ways: by reducing its structure and by reducing the number of parameters. The first way is to reduce the number of inference rules by applying rule labels. To describe the structure reduction, we propose the following definition.

Definition 2 The *structure reduction* (S_r) of the T–S model is defined as

$$S_r = \frac{z_l}{K} \tag{22}$$

where $S_r \in [0, 1]$, z_l is the number of zero-valued labels of the rules, and $K > 0$ is the number of rules.

Structure reduction S_r is an index that expresses the ratio of the number of zero-labeled rules (i.e., inactive rules) to the number of all rules. The greater its value, the greater the reduction in the structure (i.e., the simplification of the system). It is a discrete index because it is the ratio of two integers. Theoretically, it can take values from 0 (all rules are active) to 1 (no rules are active). Practically, however, at least one rule should be active in the system. Hence, the highest value that this index can take is $(K-1)/K$.

The second way to simplify the system is to reduce the number of parameters in the then-part using elastic net regression. This regression gives sparse models, which means that some coefficients might have values equal to zero. We propose the following definition to describe the sparsity.

Definition 3 The *sparsity* (S) of the T–S system is calculated as

$$S = \frac{z_c}{K(2D+1)} \tag{23}$$

where $S \in [0, 1]$, z_c is the number of polynomial coefficients equal to zero in the then-part of rules, D is the polynomial degree, and K is the number of rules.

The index S expresses the ratio of the number of zero polynomial coefficients to the number of all coefficients. The greater its value, the greater the system sparsity. It is a discrete index because it is the ratio of two integers. It can take values from 0 (all coefficients are non-zero) to 1 (all coefficients are zero).

Using the above definitions, the best T–S model is selected by minimizing an objective function in which the aim is to obtain the training error (20) and validation error (21) as small as possible and the structure reduction (22) and sparsity (23) as large as possible:

$$Q = \alpha(\mathrm{RMSE}_t + \mathrm{RMSE}_v) + (1-\alpha)(2 - S_r - S) \tag{24}$$

where $\alpha \in [0, 1]$. The quality index (24) expresses a compromise between the model accuracy and its simplification. For $\alpha = 0$, the first term of the sum (24) is equal to zero, and then, $Q = 2 - S_r - S$; that is, only simplifying the system is preferred. In this case, the value of the index Q is in the range $[1-(K-1)/K, 2]$, because S_r is in the range $[0, (K-1)/K]$, while S in [0, 1]. For $\alpha = 1$, the second term of the sum is zeroed, and then, $Q = \mathrm{RMSE}_t + \mathrm{RMSE}_v$; that is, system accuracy is preferred. In this case, the value of the index Q is greater than or equal to zero. An upper limit cannot be specified as it depends on the specific application. For $\alpha \in (0, 1)$, there is a trade-off between simplification and system accuracy. The choice of the α value depends on the application and is left to the system designer.

6.2 Design procedure

In this paper, we use one non-sparse method represented by the ANFIS model and four sparse methods utilizing the PSO, SA, GA, and PS algorithms. These methods are referred to as PSO-SR, SA-SR, GA-SR, and PS-SR, respectively. Identification of fuzzy time series models takes place in two main phases: in the first phase, fuzzy sets and rule labels are proposed by optimization algorithms, and the coarse values of the polynomial coefficients are calculated by the ridge regression. The ridge regression is used here because it is called within the objective function of the optimization algorithms and, therefore, the calculation of polynomial coefficients is expected to be fast. The main task of this regression is to avoid a situation in which the matrix $\mathbf{X}^T\mathbf{X}$ becomes singular. The regularization parameter λ is selected in our application through the trial-and-error method, but it can also be placed in the agent vector to be determined using one of the considered optimization algorithms PSO, SA, GA, or PS. Of course, the use of these algorithms does not guarantee to find the optimal values, but at most satisfactory ones. In the second phase, exact training takes place where, for the proposed sets, the final values of the coefficients are determined using elastic net regression. Using this regression makes some polynomial coefficients equal to zero, thus obtaining a sparse model (the measure of this is index S_r).

The detailed description of the design procedure presented in Fig. 2 is as follows. First, the Gaussian fuzzy sets

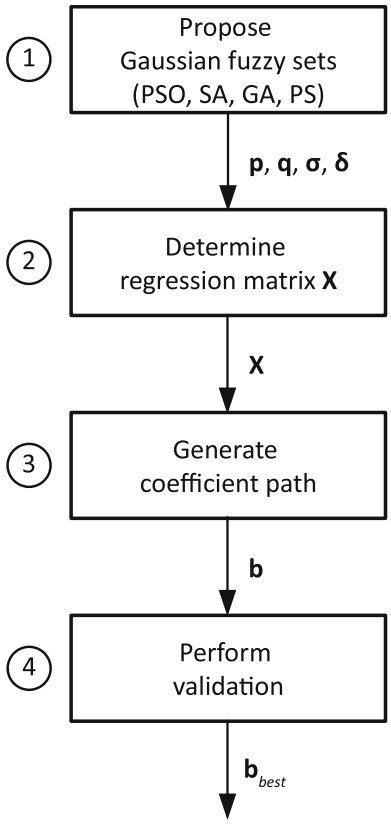

Fig. 2 Design procedure for training sparse fuzzy models

are proposed in Block 1. In the PSO-SR, SA-SR, GA-SR, and PS-SR methods, 10 propositions are created by the PSO, SA, GA, or PS algorithms. At this stage, the regression matrix $\mathbf{X}$ for the proposed fuzzy sets is generated and the polynomial coefficients are calculated using ridge regression. The outputs of Block 1 are composed of the vectors of peaks $\mathbf{p} = [p_z]$, $\mathbf{q} = [q_z]$, and the vectors of widths $\boldsymbol{\sigma} = [\sigma_z]$, $\boldsymbol{\delta} = [\delta_z]$. In Block 2, the matrix $\mathbf{X}$ is calculated, which is generated for all propositions of membership functions obtained in the previous step. This matrix is calculated using a part of the observations, called a training set. In Block 3, the coefficient path for elastic net regression is generated. In this step, we obtain many solutions, which form a set of vectors $\mathbf{b}$. In Block 4, all solutions obtained in the previous step are validated using a validation set of data. The validation is carried out for all coefficients in the coefficient path. Taking all solutions, the error RMSE_t, error RMSE_v, structure reduction S_r, sparsity S, and proposed objective function Q are calculated. After that, the smallest value of Q is determined, and, in this way, we obtain the best vector of coefficients $\mathbf{b}_{best}$.

7 Implementation

All proposed models are implemented in MATLAB with additional toolboxes. The ridge regression (15) is implemented as a custom function. The ANFIS models are implemented using the following functions from the Fuzzy Logic Toolbox [23]: `genfisOptions`, `anfisOptions`, and `anfis`. The function `genfisOptions` is used to create options for the initial model, such as the grid partitioning generation method, the number and type of input membership functions, and the output membership function type. The number of epochs and the validation data are assigned using the function `anfisOptions`. The ANFIS models are trained using the function `anfis` with the training data and the specified options as arguments.

The sparse regressions are implemented using the function `elasticnet` from the toolbox SpaSM [27]. This function takes as arguments the regression matrix $\mathbf{X}$ and vector $\mathbf{y}$. Additionally, the function `elasticnet` has δ and λ, which are the regularization parameters. As a result, this function returns the coefficient path as a set of propositions of $\mathbf{b}$ from which the best solution is chosen. The optimization algorithms are implemented using the methods from the Global Optimization Toolbox [24]. From this toolbox, the following functions are utilized: `particleswarm` for PSO, `simulannealbnd` for SA, `ga` for GA, and `patternsearch` for PS. These functions allow the solution to be determined using the user-defined bounds. They use the objective function (24) as one of the arguments and operate on a vector that contains the if-part parameters and the labels of the rules, as presented in Fig. 3, where p_z, q_z are the peaks, σ_z, δ_z are the widths, $z = 1, \ldots, Z$, Z is the number of fuzzy sets for the inputs, $l_1, \ldots, l_K$ are the labels of the rules, and K is the number of rules. It should be added that, in this paper, the regularization parameters λ for ridge regression and δ, λ for elastic net regression are searched by trial-and-error method. Another solution is also possible, consisting in placing them in the agent and searching for their values using one of the considered optimization algorithms PSO, SA, GA, or PS.

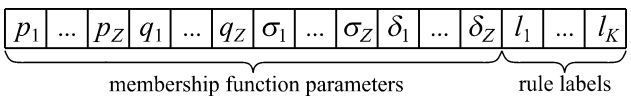

Fig. 3 Structure of the agent for global optimization methods; p_z, q_z are the peaks, σ_z, δ_z are the widths, where $z = 1, \ldots, Z$, Z is the number of fuzzy sets for the inputs, $l_1, \ldots, l_K$ are the labels of the rules, and K is the number of rules

8 Experimental results

This section describes the results of experiments in which the proposed methods are used for identifying time series models. These methods are compared with the well-known ANFIS model [14]. In all experiments, the following parameters are used. For the inputs of fuzzy systems, three Gaussian fuzzy sets are defined, which yields nine fuzzy inference rules. In Experiments 1–3, the fuzzy systems are of the first order; that is, the then-part functions are linear. It results from the fact that in the ANFIS model, the then-part of rules can contain constant or first-order functions. In Experiment 4, we consider the T–S fuzzy system of orders 1–5. The number of evaluations of the objective function for the PSO, SA, GA, and PS methods is set to 3000, and other parameters of these methods have default values.

8.1 Experiment 1

The Box–Jenkins gas furnace dataset [13, 21, 34] consists of 296 pairs $[u(t), y(t)]$ of input–output observations recorded from a laboratory furnace with the sampling interval of nine seconds. The input $u(t)$ is the methane flow rate, and the output is the percentage of carbon dioxide (CO_2) concentration in the off-gas. The goal is to build a fuzzy model to predict $y(t)$ using these data. The signals $u(t-3)$ and $y(t-1)$ are chosen as the model inputs, and therefore, the structure of the model is expressed as

$$y(t) = f(u(t-3), y(t-1)) \tag{25}$$

The first 150 data pairs are used as the training set, and the other pairs are used as the validation set.

In training the fuzzy models, the input $u(t-3)$ is bounded in the interval $[p_1, p_Z] = [-2.716, 2.834]$, while the input $y(t-1)$ is bounded in the interval $[q_1, q_Z] = [45.60, 60.50]$. The widths of fuzzy sets are bounded in the intervals $[\sigma_{min}, \sigma_{max}] = [0.2357, 5.892]$ and $[\delta_{min}, \delta_{max}] = [0.6327, 15.82]$. The regularization parameters are $\lambda = 1e{-}01$ for ridge regression (15) and $\delta = 1e{-}06$ for ENET regression (16). The parameter in the quality criterion (24) is $\alpha = 0.67$.

8.1.1 ANFIS model

In training the ANFIS model, the number of epochs can be specified. Figure 4 shows the dependence of the validation error on the number of training epochs. The minimum of this error occurs at epoch 54. Choosing the number of epochs after this point indicates the overfitting of the model parameters to the training data. Therefore, epoch 54 is chosen to obtain the best generalization performance.

The results for the obtained ANFIS model are presented in Table 2. The model has training error $\mathrm{RMSE}_t = 0.2142$ and validation error $\mathrm{RMSE}_v = 0.5391$. The fuzzy rules presented in Table 3 can be written as

Table 2 Performance comparison of T–S systems for Experiment 1; RMSE_t is the training error, RMSE_v is the validation error, S_r is the structure reduction, S is the sparsity, and Q is the value of objective function

Alg.	RMSE_t	RMSE_v	S_r	S	Q
ANFIS	0.2142	0.5391	0	0	1.190
PSO-SR	0.2524	0.5187	0.2222	0.7037	**0.8711**
SA-SR	0.2661	0.5444	0.2222	0.7037	0.8975
GA-SR	0.2455	0.5197	0.1111	0.7037	0.9038
PS-SR	0.2406	0.5110	0	0.4815	1.005

The best result is marked in bold font

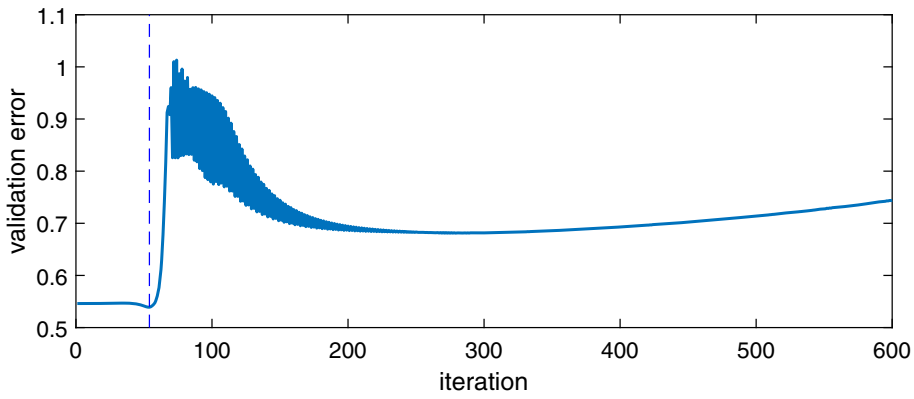

Fig. 4 Experiment 1: Validation error for the ANFIS model

Table 3 Parameters of T–S systems in Experiment 1; p, q, σ, δ are the parameters of membership functions in the if-part of fuzzy rules, and v_1, w_1, c are the polynomial coefficients in the then-part

Rule	Label	p	σ	q	δ	v_1	w_1	c
ANFIS								
Rule 1	1	− 2.382	1.086	45.65	3.240	− 1.092	− 2.519	172.3
Rule 2	1	− 2.382	1.086	52.92	3.048	− 0.7758	0.3276	37.14
Rule 3	1	− 2.382	1.086	60.56	2.814	− 0.8038	0.5630	24.96
Rule 4	1	0.5643	0.7287	45.65	3.240	− 0.4673	0.3954	29.34
Rule 5	1	0.5643	0.7287	52.92	3.048	− 1.085	0.3869	32.84
Rule 6	1	0.5643	0.7287	60.56	2.814	8.556	0.9031	24.32
Rule 7	1	2.723	1.075	45.65	3.240	− 1.182	0.6467	18.72
Rule 8	1	2.723	1.075	52.92	3.048	− 0.5927	0.4416	27.64
Rule 9	1	2.723	1.075	60.56	2.814	238.4	− 4.260	− 8.838
PSO-SR								
Rule 1	0	− 2.701	0.2357	45.68	3.783	0	0	0
Rule 2	1	− 2.701	0.2357	48.04	4.076	0	1.070	0
Rule 3	1	− 2.701	0.2357	60.50	3.844	0	1.051	0
Rule 4	0	− 2.716	0.7576	45.68	3.783	0	0	0
Rule 5	1	− 2.716	0.7576	48.04	4.076	0	1.354	0
Rule 6	1	− 2.716	0.7576	60.50	3.844	0	1.014	0
Rule 7	1	− 1.572	5.892	45.68	3.783	0	1.096	0
Rule 8	1	− 1.572	5.892	48.04	4.076	− 1.713	0.9763	0
Rule 9	1	− 1.572	5.892	60.50	3.844	0	0.9883	0

$$
\begin{aligned}
&\text{Rule 1 : If } x_1 \text{ is } g(x_1, -2.382, 1.086)\\
&\qquad \text{and } x_2 \text{ is } g(x_2, 45.65, 3.240)\\
&\qquad \text{then } y = -1.092x_1 - 2.519x_2 + 172.3\\
&\qquad \ldots\\
&\text{Rule 9 : If } x_1 \text{ is } g(x_1, 2.723, 1.075)\\
&\qquad \text{and } x_2 \text{ is } g(x_2, 60.56, 2.814)\\
&\qquad \text{then } y = 238.4x_1 - 4.260x_2 - 8.838
\end{aligned}
\tag{26}
$$

where $x_1 = u(t-3)$ and $x_2 = y(t-1)$. No rule is removed from the inference system, and there are no coefficients equal to zero in the then-part of fuzzy rules. Therefore, the structure reduction (22) and sparsity (23) are equal to zero. The quality index (24) for the ANFIS model is 1.190.

8.1.2 Sparse fuzzy models

The results for sparse models are presented in Table 2. The smallest value of the objective function Q is equal to 0.8711, which is obtained by the PSO-SR method. For this method, the training error is $\text{RMSE}_t = 0.2524$, and the validation error is $\text{RMSE}_v = 0.5187$. RMSE_v is smaller than that for the ANFIS model. The structure reduction S_r is 0.2222, which means that the PSO-SR method removes

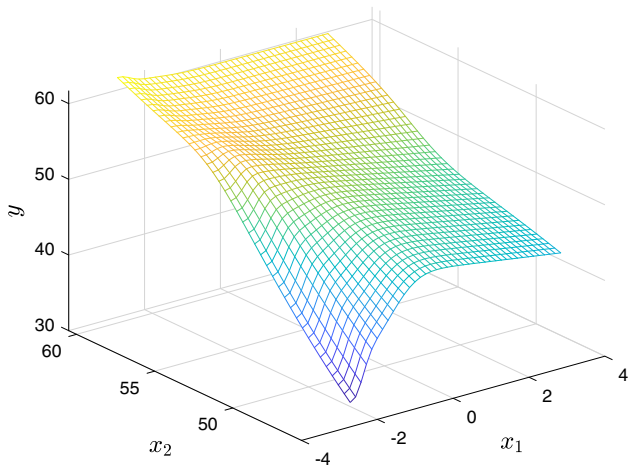

Fig. 5 Experiment 1: Fuzzy inference system's output surface for the model obtained by the PSO-SR method

22% of the nine rules. The sparsity S is 0.7037, which means that this method zeroes out 70% of the 27 polynomial coefficients. Figure 5 shows the output surface, and Table 3 lists the parameters of the T–S system calculated by the PSO-SR method. Based on this parameters, the fuzzy inference rules for the PSO-SR model can be written as

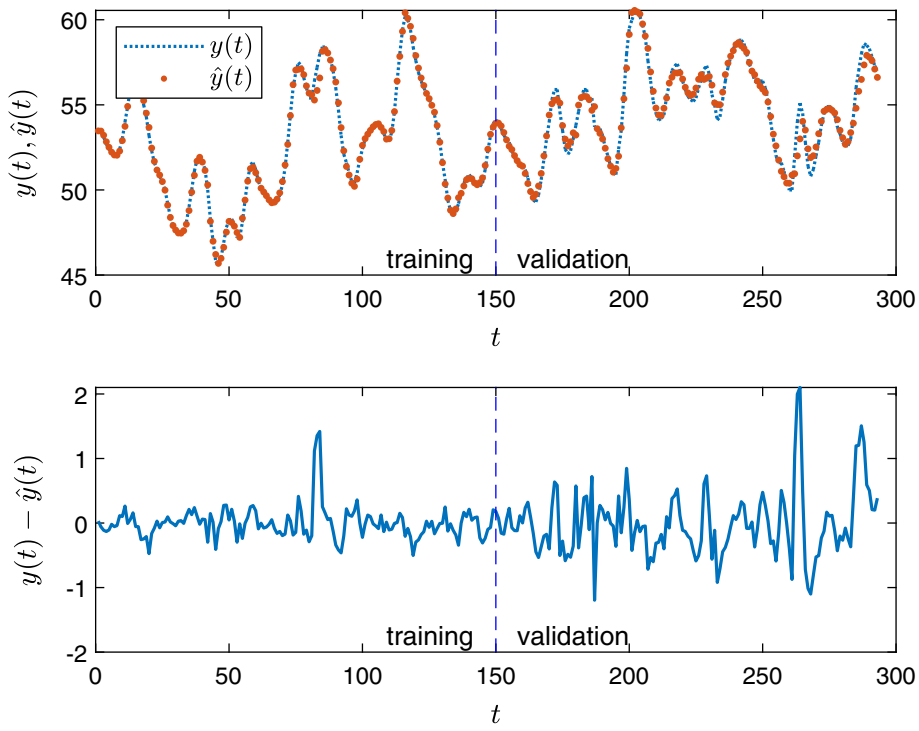

Fig. 6 Experiment 1: Comparison of the real $y(t)$ and predicted $\hat{y}(t)$ values for the model obtained by the PSO-SR method

$$
\begin{aligned}
&\text{Rule 2}: \text{If } x_1 \text{ is } g(x_1, -2.701, 0.2357)\\
&\qquad \text{and } x_2 \text{ is } g(x_2, 48.04, 4.076)\\
&\qquad \text{then } y = 1.070x_2\\
&\text{Rule 3}: \text{If } x_1 \text{ is } g(x_1, -2.701, 0.2357)\\
&\qquad \text{and } x_2 \text{ is } g(x_2, 60.50, 3.844)\\
&\qquad \text{then } y = 1.051x_2\\
&\text{Rule 5}: \text{If } x_1 \text{ is } g(x_1, -2.716, 0.7576)\\
&\qquad \text{and } x_2 \text{ is } g(x_2, 48.04, 4.076)\\
&\qquad \text{then } y = 1.354x_2\\
&\quad \ldots\\
&\text{Rule 9}: \text{If } x_1 \text{ is } g(x_1, -1.572, 5.892)\\
&\qquad \text{and } x_2 \text{ is } g(x_2, 60.50, 3.844)\\
&\qquad \text{then } y = 0.9883x_2
\end{aligned}
\tag{27}
$$

Figure 9 shows the real value y, the predicted value $\hat{y}$, and the error $y - \hat{y}$ for the model obtained using the PSO-SR method.

8.2 Experiment 2

This experiment uses the hair dryer data collected from a process described in [23, 25]. In this process, the air is heated at the tube inlet using a resistor wire, similar to a hair dryer. The input is the voltage applied to the heater, and the output is the air temperature at the tube outlet. The input–output data points were collected from the process, with the input changing between 3.41 and 6.41 V. The data points are collected at a sampling time of 0.08 s. To identify the fuzzy models, the dataset is divided into a training set containing the first 300 points and a validation set containing the remaining 300 points. The structure of the model is

$$y(t) = f(y(t-1), u(t-1)) \tag{28}$$

where $y(t-1)$, $u(t-1)$ are the inputs, and $y(t)$ is the output.

While training, the inputs are bounded in the intervals $[p_1, p_Z] = [3.201, 6.192]$ and $[q_1, q_Z] = [3.410, 6.410]$. The widths of the fuzzy sets are bounded in the intervals $[\sigma_{min}, \sigma_{max}] = [0.1270, 3.176]$ and $[\delta_{min}, \delta_{max}] = [0.1274, 3.185]$. The regularization parameters are $\lambda = 5$ for ridge regression (15) and $\delta = 1\text{e}{-}06$ for ENET regression (16). The parameter in the quality criterion (24) is $\alpha = 0.9$.

8.2.1 ANFIS model

Figure 7 shows that the minimum validation error for the ANFIS model occurs at epoch 97. Choosing this number of epochs, we obtain the results presented in Table 4. The

Table 4 Performance comparison of T–S systems for Experiment 2; RMSE_t is the training error, RMSE_v is the validation error, S_r is the structure reduction, S is the sparsity, and Q is the value of objective function

Alg.	RMSE_t	RMSE_v	S_r	S	Q
ANFIS	4.123e−07	0.1907	0	0	0.3717
PSO-SR	0.0013	0.1906	0.4444	0.8148	0.2468
SA-SR	0.0018	0.1906	0.6667	0.8148	**0.2250**
GA-SR	0.0054	0.1910	0.3333	0.8148	0.2620
PS-SR	0.0026	0.1906	0.4444	0.7778	0.2516

The best result is marked in bold font

training error is equal to $\text{RMSE}_t = 4.123\text{e}{-07}$, and the validation error is equal to $\text{RMSE}_v = 0.1907$. The fuzzy rules are presented in Table 5. All rules are included in the inference system, and there are no zero coefficients in the then-part of fuzzy rules.

8.2.2 Sparse fuzzy models

The results presented in Table 4 show that the smallest value of Q, equal to 0.2250, is obtained for the SA-SR algorithm. For this algorithm, the training error RMSE_t is 0.0018, the validation error RMSE_v is 0.1906, the structure reduction S_r is 0.6667, and the sparsity S is 0.8148. The SA-SR method removes 67% of the nine rules and zeroes out 82% of the 27 polynomial coefficients. Figure 8 shows the output surface, and Table 5 lists the parameters of the T–S system obtained by the SA-SR algorithm. The removed rules (1, 2, 4, 5, 7, and 8) have the zero polynomial in the then-part. The three rules (3, 6, and 9) left in the system can be written as

$$\begin{aligned}
&\text{Rule 3}: \text{If } x_1 \text{ is } g(x_1, 3.383, 3.056)\\
&\qquad \text{and } x_2 \text{ is } g(x_2, 3.720, 3.159)\\
&\qquad \text{then } y = 0.7771x_1 - 0.2938x_2\\
&\text{Rule 6}: \text{If } x_1 \text{ is } g(x_1, 6.091, 3.106)\\
&\qquad \text{and } x_2 \text{ is } g(x_2, 3.720, 3.159)\\
&\qquad \text{then } y = 1.505x_1 + 0.9747x_2\\
&\text{Rule 9}: \text{If } x_1 \text{ is } g(x_1, 6.047, 2.789)\\
&\qquad \text{and } x_2 \text{ is } g(x_2, 3.720, 3.159)\\
&\qquad \text{then } y = 0.2138x_1
\end{aligned} \tag{29}$$

Figure 9 shows the real value y, the predicted value $\hat{y}$, and the error $y - \hat{y}$ for the model obtained by the SA-SR method.

8.3 Experiment 3

The goal of this experiment is to predict the time series generated by the Mackey–Glass chaotic system. This benchmark problem is frequently used in testing neural

Table 5 Parameters of T–S systems in Experiment 2; p, q, σ, δ are the parameters of membership functions in the if-part of fuzzy rules, and v_1, w_1, c are the polynomial coefficients in the then-part

Rule	Label	p	σ	q	δ	v_1	w_1	c
ANFIS								
Rule 1	1	3.201	0.6291	3.410	0.6370	1	8.309e−06	2.437e−06
Rule 2	1	3.201	0.6291	4.909	0.3345	8.600e−05	5.433e−10	1.344e−10
Rule 3	1	3.201	0.6291	6.410	0.6370	1	4.275e−06	6.670e−07
Rule 4	1	4.682	0.6291	3.410	0.6370	1	1.051e−05	3.082e−06
Rule 5	1	4.682	0.6291	4.909	0.3345	8.600e−05	6.496e−10	1.641e−10
Rule 6	1	4.682	0.6291	6.410	0.6370	1	4.527e−06	7.066e−07
Rule 7	1	6.163	0.6291	3.410	0.6370	1	1.853e−05	5.435e−06
Rule 8	1	6.163	0.6291	4.909	0.3345	8.600e−05	1.086e−09	2.803e−10
Rule 9	1	6.163	0.6291	6.410	0.6370	1	6.572e−06	1.025e−06
SA-SR								
Rule 1	0	3.383	3.056	5.570	0.3915	0	0	0
Rule 2	0	3.383	3.056	4.218	0.1374	0	0	0
Rule 3	1	3.383	3.056	3.720	3.159	0.7771	− 0.2938	0
Rule 4	0	6.091	3.106	5.570	0.3915	0	0	0
Rule 5	0	6.091	3.106	4.218	0.1374	0	0	0
Rule 6	1	6.091	3.106	3.720	3.159	1.505	0.9747	0
Rule 7	0	6.047	2.789	5.570	0.3915	0	0	0
Rule 8	0	6.047	2.789	4.218	0.1374	0	0	0
Rule 9	1	6.047	2.789	3.720	3.159	0.2138	0	0

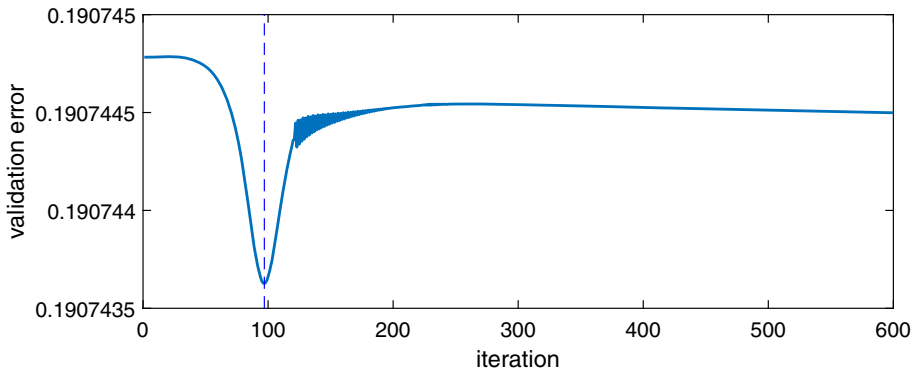

Fig. 7 Experiment 2: Validation error for the ANFIS model

networks and fuzzy logic models [9, 13, 26, 28, 33]. The system is described by the following differential equation [22]:

$$\dot{y}(t) = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{30}$$

The fourth-order Runge–Kutta method with $x(0) = 1.2$ and $\tau = 17$ was used to obtain the time series [23]. The signals $y(t-18)$ and $y(t-12)$ are chosen as the model inputs, and $y(t+6)$ is chosen as the model output:

$$y(t+6) = f(y(t-18), y(t-12)) \tag{31}$$

There are 1000 input/output samples for each t ranging from 118 to 1117. The first 500 observations are used as training data and the remaining 500 observations as validation data.

While training, the inputs are bounded in the interval $[p_1, p_Z] = [q_1, q_Z] = [0.4256, 1.314]$. The widths of the fuzzy sets are bounded in the interval $[\sigma_{min}, \sigma_{max}] = [\delta_{min}, \delta_{max}] = [0.0377, 0.9428]$. The parameters are $\lambda = 1\text{e}{-}04$ for ridge regression (15) and $\delta = 1\text{e}{-}06$ for ENET regression (16). The quality criterion (24) is calculated with $\alpha = 0.9$.

8.3.1 ANFIS model

Based on Fig. 10, 1600 epochs are chosen. As shown in Table 6, the training error RMSE_t is equal to 0.0308, and the validation error RMSE_v is equal to 0.0304. The structure reduction in the ANFIS model is zero, the sparsity is zero, and the quality index Q is 0.2163.

8.3.2 Sparse fuzzy models

The performance comparison for various fuzzy models is presented in Table 6. The smallest value of Q, equal to 0.1675, is obtained using the SA-SR method. For this method, the training error RMSE_t is 0.0399, the validation error RMSE_v is 0.0392, the structure reduction S_r is 0.3333, and the sparsity S is 0.7037. The SA-SR method removes 33% of the nine rules and zeroes out 70% of the 27 polynomial coefficients. Figure 11 shows the output surface, and Table 7 presents the fuzzy system parameters obtained by the SA-SR method. The fuzzy rules can be written as

$$\begin{aligned}
&\text{Rule 1}: \text{If } x_1 \text{ is } g(x_1, 1.094, 0.8363)\\
&\quad \text{and } x_2 \text{ is } g(x_2, 0.4462, 0.1942)\\
&\quad \text{then } y = -0.4938\\
&\text{Rule 2}: \text{If } x_1 \text{ is } g(x_1, 1.094, 0.8363)\\
&\quad \text{and } x_2 \text{ is } g(x_2, 1.209, 0.6497)\\
&\quad \text{then } y = -1.865x_2\\
&\text{Rule 4}: \text{If } x_1 \text{ is } g(x_1, 1.185, 0.2265)\\
&\quad \text{and } x_2 \text{ is } g(x_2, 0.4462, 0.1942)\\
&\quad \text{then } y = -0.5587\\
&\text{Rule 5}: \text{If } x_1 \text{ is } g(x_1, 1.185, 0.2265)\\
&\quad \text{and } x_2 \text{ is } g(x_2, 1.209, 0.6497)\\
&\quad \text{then } y = -9.160x_1 + 6.293x_2\\
&\text{Rule 6}: \text{If } x_1 \text{ is } g(x_1, 1.185, 0.2265)\\
&\quad \text{and } x_2 \text{ is } g(x_2, 0.7474, 0.7001)\\
&\quad \text{then } y = 3.100\\
&\text{Rule 9}: \text{If } x_1 \text{ is } g(x_1, 1.009, 0.7578)\\
&\quad \text{and } x_2 \text{ is } g(x_2, 0.7474, 0.7001)\\
&\quad \text{then } y = 2.588x_1 + 4.663
\end{aligned} \tag{32}$$

Figure 12 shows the real value y, predicted value $\hat{y}$, and error $y - \hat{y}$ for the chosen model.

8.4 Experiment 4

This experiment provides an example of using high-order T–S models for predicting the time series. The dataset and

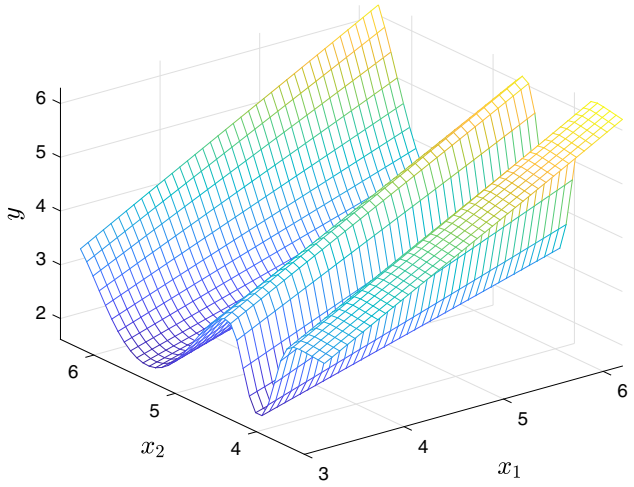

Fig. 8 Experiment 2: Fuzzy inference system's output surface for the model obtained by the SA-SR method

Table 6 Performance comparison of T–S systems for Experiment 3; RMSE_t is the training error, RMSE_v is the validation error, S_r is the structure reduction, S is the sparsity, and Q is the value of objective function

Alg.	RMSE_t	RMSE_v	S_r	S	Q
ANFIS	0.0308	0.0304	0	0	0.2163
PSO-SR	0.0401	0.0340	0.1111	0.7037	0.1906
SA-SR	0.0399	0.0392	0.3333	0.7037	**0.1675**
GA-SR	0.0418	0.0415	0.2222	0.6296	0.1898
PS-SR	0.0335	0.0331	0.2222	0.2593	0.2118

The best result is marked in bold font

parameters are the same as in Experiment 3. We consider the ANFIS model of first order and the PSO-SR model with order changing from one to five. The experimental results

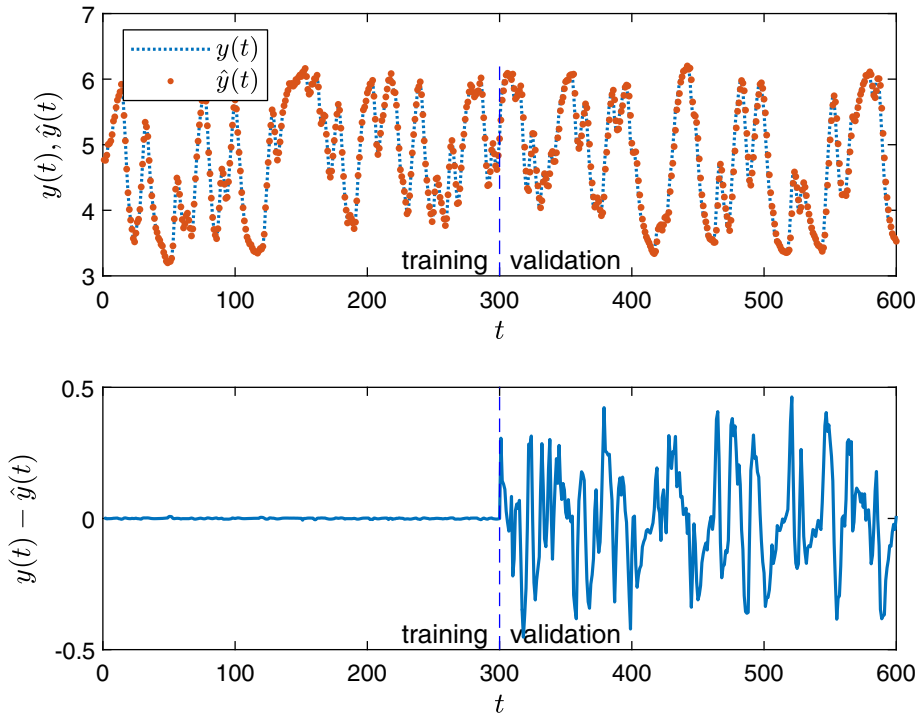

Fig. 9 Experiment 2: Comparison of the real $y(t)$ and predicted $\hat{y}(t)$ values for the model obtained by the SA-SR method

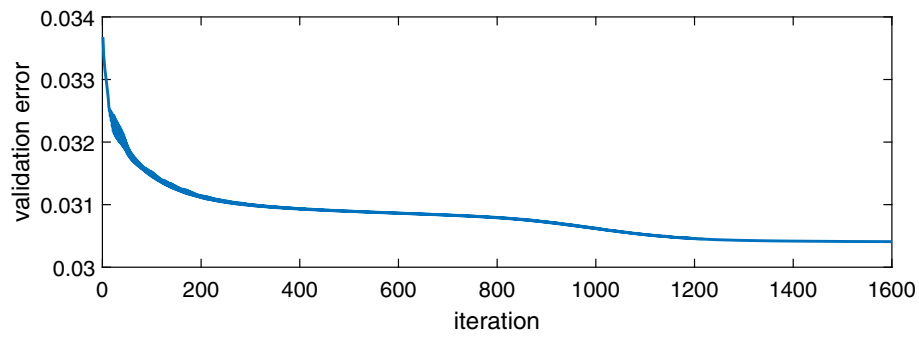

Fig. 10 Experiment 3: Validation error for the ANFIS model

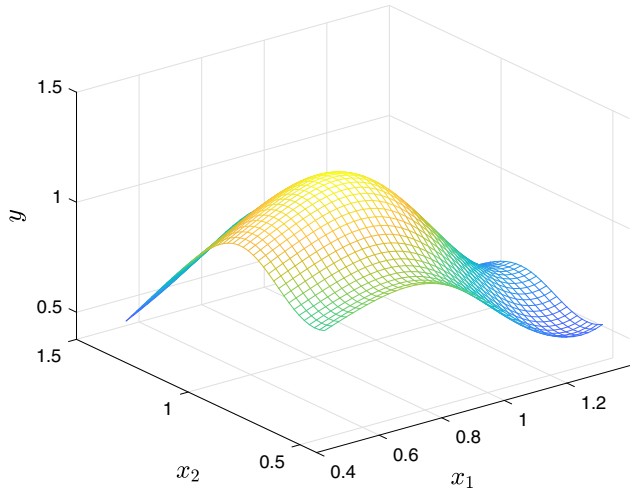

Fig. 11 Experiment 3: Fuzzy inference system's output surface the model obtained by the SA-SR method

are presented in Table 8. For all cases, the quality index Q for the PSO-SR model is better than that for the ANFIS model, and that for the fifth-order system has the lowest value equal to 0.1321. This example shows that using a higher-order system can improve the model performance.

9 Conclusions

A proposition of hybrid methods that combines a sparse regression, rule labels, and a global optimization method to identify Takagi–Sugeno fuzzy models for time series predictions is described. The if-part parameters of fuzzy rules are determined by one of the following optimization methods: particle swarm optimization, simulated annealing, genetic algorithm, or pattern search. The then-part parameters are determined by ridge regression and elastic net regression. A new quality criterion that presents a compromise between the accuracy of a model and its simplification is proposed. The simplification is based on reducing the number of polynomial parameters in the then-part and removing unnecessary rules by using labels. The well-known adaptive neuro-fuzzy inference system is used to compare the results. The experimental results show that the applied methods can improve the fuzzy models by zeroing some of their coefficients and removing the unnecessary rules. Moreover, in all conducted experiments, the proposed methods improve the result obtained using the reference method.

However, the proposed methods have some limitations in terms of the size data size. We consider datasets that are commonly used to test time series prediction algorithms. The size of these sets is appropriate for using ridge and elastic net regressions, which are batch methods. This means that all data are used to compute the regression

Table 7 Parameters of T–S systems in Experiment 3; p, q, σ, δ are the parameters of membership functions in the if-part of fuzzy rules, and v_1, w_1, c are the polynomial coefficients in the then-part

Rule	Label	p	σ	q	δ	v_1	w_1	c
ANFIS								
Rule 1	1	0.3829	0.2935	0.7244	0.1965	0.1369	2.464	0.2068
Rule 2	1	0.3829	0.2935	0.7120	0.2543	1.860	− 1.661	0.4616
Rule 3	1	0.3829	0.2935	1.347	0.0792	23.48	6.009	− 16.17
Rule 4	1	0.9719	0.0449	0.7244	0.1965	− 14.89	5.645	11.59
Rule 5	1	0.9719	0.0449	0.7120	0.2543	9.946	− 2.534	− 6.980
Rule 6	1	0.9719	0.0449	1.347	0.0792	− 33.50	36.29	− 8.941
Rule 7	1	1.149	0.2828	0.7244	0.1965	0.4291	− 2.256	3.875
Rule 8	1	1.149	0.2828	0.7120	0.2543	− 2.010	3.209	− 1.218
Rule 9	1	1.149	0.2828	1.347	0.0792	0.9441	0.6491	− 1.856
SA-SR								
Rule 1	1	1.094	0.8363	0.4462	0.1942	0	0	− 0.4938
Rule 2	1	1.094	0.8363	1.209	0.6497	0	− 1.865	0
Rule 3	0	1.094	0.8363	0.7474	0.7001	0	0	0
Rule 4	1	1.185	0.2265	0.4462	0.1942	0	0	− 0.5587
Rule 5	1	1.185	0.2265	1.209	0.6497	− 9.160	6.293	0
Rule 6	1	1.185	0.2265	0.7474	0.7001	0	0	3.100
Rule 7	0	1.009	0.7578	0.4462	0.1942	0	0	0
Rule 8	0	1.009	0.7578	1.209	0.6497	0	0	0
Rule 9	1	1.009	0.7578	0.7474	0.7001	2.588	0	4.663

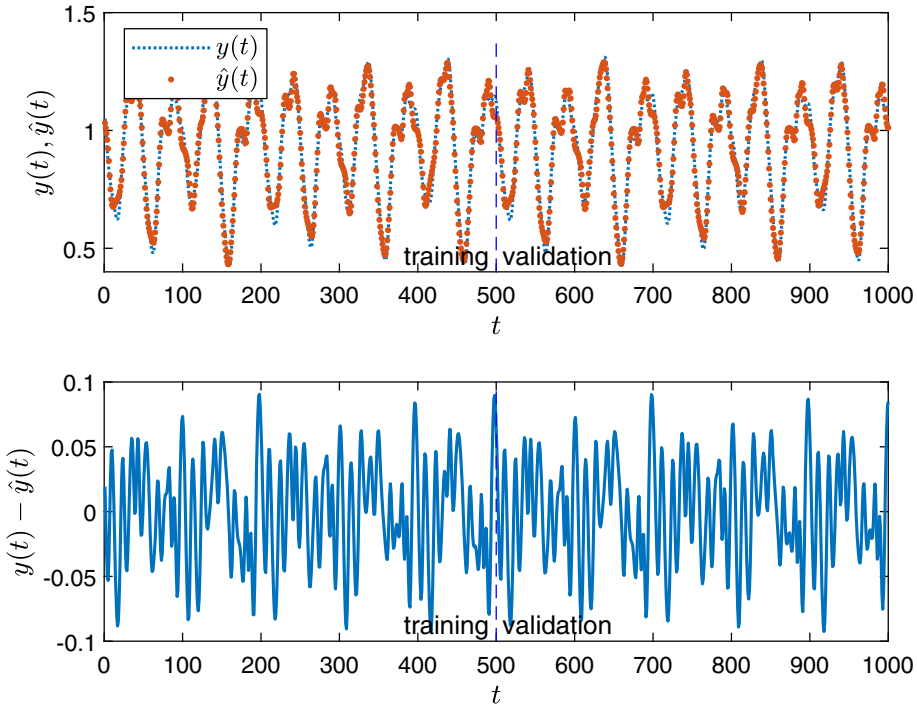

Fig. 12 Experiment 3: Comparison of the real $y(t)$ and predicted $\hat{y}(t)$ values for the model obtained by the SA-SR method

Table 8 Performance comparison of T–S systems for Experiment 4; RMSE_t is the training error, RMSE_v is the validation error, S_r is the structure reduction, S is the sparsity, and Q is the value of objective function

Alg.	Order	RMSE_t	RMSE_v	S_r	S	Q
ANFIS	1	0.0308	0.0304	0	0	0.2163
PSO-SR	1	0.0401	0.0340	0.1111	0.7037	0.1906
PSO-SR	2	0.0444	0.0435	0.4444	0.7778	0.1568
PSO-SR	3	0.0395	0.0390	0.3333	0.7619	0.1611
PSO-SR	4	0.0344	0.0340	0.3333	0.8148	0.1467
PSO-SR	5	0.0414	0.0403	0.5556	0.8586	**0.1321**

The best result is marked in bold font

matrix. As a result, the ridge regression applied in a objective function of the optimization algorithms is very fast. Unfortunately, for large datasets, there is a limitation related to the size of the regression matrix in MATLAB. This size is the product of the number of observations (number of rows) and the number of predictors (number of columns), and its upper limit depends on the computer and the system used. In this case, recursive regressions should be applied. We plan to consider such a solution in the future.

Funding No funding was received for conducting this study.

Data availability All experiments were carried out on publicly available datasets.

Code availability The research used generally available functions and toolboxes of the MATLAB program.

Declarations

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

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication All authors consented to the publication of the research.

References

1. Aladi JH, Wagner C, Garibaldi JM (2016) A simplified method of FOU design utlising simulated annealing. In: Proceedings—2015 IEEE international conference on systems, man, and cybernetics, SMC 2015. pp 2255–2261. https://doi.org/10.1109/SMC.2015.394 (2016)
2. Almaraashi M, John R, Coupland S, Hopgood A (2010) Time series forecasting using a TSK fuzzy system tuned with simulated annealing. In: 2010 IEEE world congress on computational intelligence, WCCI 2010. https://doi.org/10.1109/FUZZY.2010.5584523
3. Azad A, Pirayesh J, Farzin S, Malekani L, Moradinasab S, Kisi O (2019) Application of heuristic algorithms in improving performance of soft computing models for prediction of min, mean and max air temperatures. Eng J 23(6):83–98. https://doi.org/10.4186/ej.2019.23.6.83
4. Chen SM, Hsin WC (2015) Weighted fuzzy interpolative reasoning based on the slopes of fuzzy sets and particle swarm optimization techniques. IEEE Trans Cybern 45(7):1250–1261. https://doi.org/10.1109/TCYB.2014.2347956
5. Eberhart RC, Shi Y (2000) Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of the 2000 congress on evolutionary computation 2000, vol 1, pp 84–88. https://doi.org/10.1109/CEC.2000.870279
6. Efron B, Hastie T, Johnstone I, Tibshirani R (2004) Least angle regression. Ann Stat 32(2):407–499. https://doi.org/10.1214/009053604000000067
7. Gaxiola F, Melin P, Valdez F, Castro JR, Castillo O (2016) Optimization of type-2 fuzzy weights in backpropagation learning for neural networks using GAs and PSO. Appl Soft Comput J 38:860–871. https://doi.org/10.1016/j.asoc.2015.10.027
8. Glover FW, Kochenberger GA (2003) Handbook of metaheuristics. Springer, Berlin. https://doi.org/10.1007/978-1-4419-1665-5
9. Ho DT, Garibaldi JM (2014) Context-dependent fuzzy systems with application to time-series prediction. IEEE Trans Fuzzy Syst 22(4):778–790. https://doi.org/10.1109/TFUZZ.2013.2272645
10. Hoerl AE, Kennard RW (1970) Ridge regression: biased estimation for nonorthogonal problems. Technometrics 12(1):55–67. https://doi.org/10.1080/00401706.1970.10488634
11. Holland JH (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT Press, Cambridge
12. Hooke R, Jeeves TA (1961) "Direct search'' solution of numerical and statistical problems. J ACM 8(2):212–229. https://doi.org/10.1145/321062.321069
13. Isfahani MK, Zekri M, Marateb HR, Mañanas MA (2019) Fuzzy jump wavelet neural network based on rule induction for dynamic nonlinear system identification with real data applications. PLOS ONE 14(12):1–26. https://doi.org/10.1371/journal.pone.0224075
14. Jang JR (1993) ANFIS: adaptive-network-based fuzzy inference system. IEEE Trans Syst Man Cybern 23(3):665–685. https://doi.org/10.1109/21.256541
15. Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol 4, pp 1942–1948. IEEE Press, Piscataway, NJ. https://doi.org/10.1109/ICNN.1995.488968
16. Khosravi A, Machado L, Nunes RO (2018) Time-series prediction of wind speed using machine learning algorithms: a case study Osorio wind farm. Braz Appl Energy 224(May):550–566. https://doi.org/10.1016/j.apenergy.2018.05.043
17. 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
18. Kolda TG, Lewis RM, Torczon V (2003) Optimization by direct search: new perspectives on some classical and modern methods. SIAM Rev 45:385–482. https://doi.org/10.1137/S003614450242889
19. Li C, Hu JW (2012) A new ARIMA-based neuro-fuzzy approach and swarm intelligence for time series forecasting. Eng Appl Artif Intell 25(2):295–308. https://doi.org/10.1016/j.engappai.2011.10.005
20. Lin CJ, Jeng SY, Lin HY, Yu CY (2020) Design and verification of an interval type-2 fuzzy neural network based on improved particle swarm optimization. Appl Sci (Switzerland) 10(9):3041. https://doi.org/10.3390/app10093041
21. Lin L, Guo F, Xie X, Luo B (2015) Novel adaptive hybrid rule network based on TS fuzzy rules using an improved quantum-behaved particle swarm optimization. Neurocomputing 149(PB):1003–1013. https://doi.org/10.1016/j.neucom.2014.07.033
22. Mackey MC, Glass L (1977) Oscillation and chaos in physiological control systems. Science 197(4300):287–289. https://doi.org/10.1126/science.267326
23. MathWorks (2019) Fuzzy logic toolbox: user's guide
24. MathWorks (2019) Global optimization toolbox: user's guide
25. MathWorks (2019) System identification toolbox: user's guide
26. Singh S, Singh S, Banga VK (2020) Design of fuzzy logic system framework using evolutionary techniques. Soft Comput 24(6):4455–4468. https://doi.org/10.1007/s00500-019-04207-9
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. Soto J, Melin P, Castillo O (2018) A new approach for time series prediction using ensembles of IT2FNN models with optimization of fuzzy integrators. Int J Fuzzy Syst 20(3):701–728. https://doi.org/10.1007/s40815-017-0443-6
29. Tibshirani R (1996) Regression shrinkage and selection via the lasso. J R Stat Soc Ser B (Methodol) 58:267–288
30. Wang L, Mendel JM (1992) Fuzzy basis functions, universal approximation, and orthogonal least-squares learning. IEEE Trans Neural Netw 3(5):807–814. https://doi.org/10.1109/72.159070
31. Whitley D (1994) A genetic algorithm tutorial by Darrell Whitley. Stat Comput 4:65–85. https://doi.org/10.1007/BF00175354
32. Wiktorowicz K, Krzeszowski T (2020) Approximation of two-variable functions using high-order Takagi–Sugeno fuzzy systems, sparse regressions, and metaheuristic optimization. Soft Comput 24:1–15. https://doi.org/10.1007/s00500-020-05238-3
33. 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 J 13(5):2390–2404. https://doi.org/10.1016/j.asoc.2013.01.023
34. 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
35. 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