Soft Computing (2019) 23:11737–11757
https://doi.org/10.1007/s00500-018-03726-1

METHODOLOGIES AND APPLICATION

EEFR-R: extracting effective fuzzy rules for regression problems, through the cooperation of association rule mining concepts and evolutionary algorithms

Fatemeh Aghaeipoor[2] · Mahdi Eftekhari[1]

Published online: 5 January 2019

Abstract
Fuzzy rule-based systems, due to their simplicity and comprehensibility, are widely used to solve regression problems. Fuzzy rules can be generated by learning from data examples. However, this strategy may result in high numbers of rules that most of them are redundant and/or weak, and they affect the systems' interpretability. Hence, in this paper, a new rule learning method, EEFR-R, is proposed to extract the effective fuzzy rules from regression data samples. This method is formed through the cooperation of association rule mining concepts and evolutionary algorithms in the three stages. Indeed, the components of a Mamdani fuzzy rule-based system are generated during the first two stages, and then, they will be refined through some modifications in the last stage. In EEFR-R, fuzzy rules are extracted from numerical data using the idea of Wang and Mendel's method and utilizing the concepts of Support and Confidence; furthermore, a new rule pruning method is presented to refine these rules. By employing this method, non-effective rules can be pruned in three different modes as the preferences of a decision maker. The proposed model and its stages were validated using 19 real-world regression datasets. The experimental results and the conducted statistical tests confirmed the effectiveness of EEFR-R in terms of complexity and accuracy and in comparison with the three state-of-the-art regression solutions.

Keywords Discretization · Fuzzy rule learning · Rule pruning · Support and confidence · Particle swarm optimization

1 Introduction

In regression problems, a specified output is estimated using a set of input variables. There are a lot of methods which are employed to model regression problems, simple methods such as least squares to more advanced ones such as multi-objective evolutionary algorithms (Ratner 2017). Fuzzy inference systems (FISs) are one of the most useful methods to address regression problems. FISs, proposed based on the fuzzy set theory of Zadeh (1965, 1975), employ linguistic concepts to model input and output relationships. Since the inference ability of these systems relies on their rules, FISs are also called fuzzy rule-based systems (FRBSs). Each FRBS has a Knowledge Base (KB) which itself is comprised of two fundamental parts including Data Base (DB) and Rule Base (RB); the DB includes fuzzy set definitions and membership functions (MFs) parameters, and the RB contains a set of linguistic fuzzy If-Then rules (Riza et al. 2015). Fuzzy rules have a critical role in each FIS so that decision-making process is not possible without applying them.

There exist two types of approaches to derive fuzzy If-Then rules. In the first one, the RB is manually generated by the knowledge of human experts, while in the second one, called data-driven models, the rules are automatically extracted from numerical data by using the learning methods. The second approach is more practical in the situation of lacking human expert's knowledge or in the complex systems where a complete knowledge of the problem is not available (Riza et al. 2015). In this regard, one of the classical rule learning methods is Wang and Mendel's algorithm Wang and

Communicated by V. Loia.

✉ Mahdi Eftekhari
m.eftekhari@uk.ac.ir

Fatemeh Aghaeipoor
ff.aghaei@eng.uk.ac.ir

1 Department of Computer Engineering, Shahid Bahonar University of Kerman, Kerman, Iran

2 Department of Mathematics and Computer Science, Shahid Bahonar University of Kerman, Kerman, Iran

Mendel (1992), it obtains the rules set by learning from the training data. The simplicity and quickness of this approach have made it as a popular and widely used method, and for this reason, a lot of researchers have utilized its idea in their applications and have tried to improve it (Kato et al. 2009; Gacto et al. 2014).

In the automatic generation approaches, the number of obtained rules may become enormous, especially in the case of big datasets. Moreover, some of these generated rules may be redundant and non-effective, even they may be destructive in the cooperation with the other rules. On the other hand, the high number of rules results in losing interpretability of the fuzzy model (Alonso et al. 2015) such that the system behavior becomes hard to understand and the efficiency is highly affected. To handle these problems, there are two strategies available; the first one which relies on learning an appropriate number of effective rules from the beginning, and the second one where all possible rules are initially generated and in the second step, the non-effective rules will be pruned through an optimization process. Although the first approaches avoid exhaustive optimization tasks, the second group results in more efficient RB (Patel 2013). In this study, we attempt to combine the advantages of these two strategies. Moreover, most of the rule learning approaches perform with the same principles for different datasets with different conditions, e.g., the maximum and the minimum number of obtained rules are general and fixed parameters (Alcalá et al. 2011; Alcalá-Fdez et al. 2011a; Gacto et al. 2014). In this way, the circumstances of the problem and the preferences of the decision makers are ignored. This matter is also taken into account in the proposed rule learning method.

This paper presents EEFR-R, a new method to **E**xtract **E**ffective **F**uzzy **R**ules for **R**egression problem. Indeed, an efficient Mamdani FRBS is constructed through the cooperation of association rule mining concepts and PSO algorithm in the three stages. Input variables are preprocessed using Fayyad and Irani's discretization method, and then, all MFs are defined. In the following, the RB is constructed via a two-step process: rule generation and rule pruning. Rule generation is done using the idea of Wang and Mendel's method, and rule pruning is proposed based on integrating a PSO algorithm into a common rule mining strategy. The new rule pruning method can eliminate additional rules in three levels (modes) based on the preferences of decision makers. Finally, in the post-processing stage, an optimization algorithm is utilized to tune the MFs and adjust the rules' weight. EEFR-R is validated using 19 real-world regression datasets with different numbers of variables and samples. The performance of each stage is separately evaluated. Furthermore, the results of EEFR-R are compared with the three state-of-the-art regression solutions and some statistical tests are employed to carry out more clearly pairwise and multiple comparisons. Experimental results show the effectiveness of EEFR-R, especially in terms of complexity and accuracy.

The rest of this paper is organized as follows. Section 2 describes the preliminaries of the model. Section 3 details the three stages of EEFR-R, including preprocessing, model generation, and post-processing stage. Section 4 demonstrates the results of evaluations, comparisons, and statistical tests, and finally, Sect. 5 presents the conclusion of this study.

2 Preliminaries

The preliminaries of this study including discretization methods, association rule mining concepts, and PSO algorithm are described in this section. Before these descriptions, a review in the related literature is also done.

2.1 Literature review

There are different learning methods for the FRBS generation in the literature, e.g., learning based on space partitions, learning based on clustering approaches, learning based on gradient descent method, and learning by the means of neural networks or evolutionary algorithms (Riza et al. 2015). Learning based on space partitions, or structure-based approaches, partitions the data space using fuzzy sets and then extracts fuzzy rules based on those partitions (Liu and Cocea 2018). Cluster-based approaches do a clustering in data and then use each cluster to generate one rule (Prasad et al. 2014). The last three methods utilize the capabilities of gradient descent, neural networks, and evolutionary algorithms iteratively to learn the components of the FRBSs (Jang 1993; Fernandez et al. 2015).

One of the most successful learning algorithms for automatic generation of an FRBS is evolutionary fuzzy systems (EFSs), i.e., evolutionary algorithms have been integrated into fuzzy systems to learn or tune fuzzy elements (Fernandez et al. 2015). These hybrid systems due to their flexibility in the codification of the FRBSs, and given their ability in providing different trade-offs of accuracy and interpretability, are popular, especially in learning tasks. A rule learning process was proposed in Debie et al. (2014), it employed an evolutionary algorithm to search for attribute intervals and rules structures, simultaneously. The parameters of DB and RB can be also learned concurrently (Shill et al. 2011).

Particle swarm optimization (PSO) algorithm is one of the evolutionary algorithms which has been significantly used in FISs. Easy implementation, quick convergence, and lower complexity are prominent features of the PSO algorithm, and they have made it popular among researchers (Du and Swamy 2016). Some papers have efficiently employed these features to perform learning and tuning tasks of a fuzzy system. In Zanganeh et al. (2011), a PSO algorithm has been used to tune

```
Each entropy-based discretization algorithm performs the following steps:
  1)  Calculate total entropy of partition P
  2)  For all candidate points in P
          -  Calculate Entropy of both generated sub-partitions
          -  Find the weighted entropy*
          -  Calculate information gain*
  3)  Select point T if the information gain after accepting it, is highest
  4)  Divide P into two partitions using T
  5)  Recursively repeat step 1, 2 ,3 and 4 for both generated partitions of step 4 until the termination criteria is met

* Weighted entropy is the weighted values of entropies for two sides of a CP and it must be minimized.
* Information gain:  is the expected reduction in entropy caused by partitioning the examples according to a certain attribute.
```

Fig. 1 Pseudo-code of an entropy-based discretization algorithm (de Sá et al. 2016)

the rule's antecedent and consequent parameters with respect to the minimization of an estimated error. In another work, an efficient PSO-based approach has been proposed to construct an FRBS by using data examples (Esmin 2007). A method called fuzzy particle swarm optimization (FPSO) was proposed in Permana and Hashim (2010) to use the capabilities of PSO for generating and adjusting MF automatically. Two different fuzzy classifiers based on Mamdani and TSK FISs have been developed in Elragal (2010), where all parameters of the proposed classifiers and the structure of fuzzy rules are optimized using a PSO algorithm. Because of the faster convergence and the simplicity of PSO algorithm in comparison with genetic algorithm (it balances between exploration and exploitation in the search space Visalakshi and Sivanandam 2009), PSO algorithm is employed for the optimization tasks of this paper.

However, as mentioned in the previous section, some of the generation methods lead to an enormous number of rules. To handle this problem, there are also different strategies in the literature: strategies like removing redundant rule (Patel 2013), merging the overlapping rules (Alonso et al. 2015), rule selection (Alcalá et al. 2007), or rule pruning (Batbarai and Naidu 2014).

In some researches, the concepts of FRBSs and association rule mining are fused so that the strategists of the association rule mining can be used to refine the RB of the FRBSs (Alcalá-Fdez et al. 2011a); e.g., in Shehzad (2013), by using the concepts of Support and Confidence, a measure of significance level is defined for each rule, and then using the values of this measure, less significant rules are pruned, or in Antonelli et al. (2017), to manage unbalanced data in a fuzzy classifier, the rules' weights are defined using the scaled Support and Confidence. Due to the results of these researches, in this study, an idea related to the association rule mining concepts is also adapted and combined with a PSO algorithm to propose a new rule pruning method.

2.2 Discretization method: Fayyad and Irani's

In some machine learning applications, discretization methods are required to handle data with continuous attributes (Zeinalkhani and Eftekhari 2014). These methods are employed to convert continuous variables into discrete ones. Indeed, through a discretization process, the domain of a continuous variable is partitioned into several intervals, and consequently, a set of cut points is generated.

The term of cut point (CP) is applied for a value within the domain of a certain continuous variable so that it divides the variable's domain into two sub-partitions; one partition is less than or equal to that CP, and the other partition is greater than it. Given these definitions, if k CPs are chosen in the domain of a continuous variable, $k + 1$ partitions will be built in that domain (Garcia et al. 2013).

Each value within the variable's domain is a candidate to choose as a CP. The best CPs are picked out based on a splitting measure which is a criterion to evaluate different candidate points. Different splitting measures and accordingly different discretization methods are available in the literature, methods such as binning-based, Chi-square-based, entropy-based, and wrapper-based (Dash et al. 2011).

Entropy is one of the most commonly used discretization measures; it refers to the measure of uncertainty in information being processed. A lot of discretization algorithms have been developed using the entropy measure. They evaluate the entropy of the candidate partitions to select the CPs. Figure 1 shows a pseudo-code of a typical entropy-based discretization algorithm. The process of choosing suitable CPs starts by considering one big partition containing all values of a variable; then, among all candidate points within this partition, that point is accepted as a CP which has the highest information gain; this process is recursively repeated for each generated sub-partition until a stopping condition is met (de Sá et al. 2016).

Fayyad and Irani's is a popular entropy-based discretization method. It considers the midpoints between each pair of the accepted CPs as the candidate points; then, it evaluates all candidate points and selects that point for which the entropy is minimal. These evaluations recursively continue until a stopping condition, which is determined based on the minimal description length principle, is met. More detail about this method is available in Fayyad and Irani (1993). Fayyad and Irani's discretization method is utilized in Sect. 3.1 to prepare regression data before model generation.

2.3 Association rule mining concepts

In order to modify conflicting rules in the rule generation process of Sect. 3.2.2, and also to propose a new rule pruning method in Sect. 3.2.3, two rule evaluation metrics related to the association rule mining concepts, namely Support and Confidence, are utilized. In this section, we briefly describe them.

Support is a measure that used to calculate the frequency of a certain rule in a rules set, while Confidence is a measure that employed to specify the reliability of the rules (Bhargava and Shukla 2016). To define these measures, first of all, we denote a generic fuzzy If-Then rule (it is defined in Eq. (18) in "Appendix A") as:

$$\text{Rule}^i : A_i \longrightarrow B_i \tag{1}$$

in which $A_i = \{A_i^1,\ A_i^2,\ \ldots,\ A_i^n\}$ and it includes all fuzzy sets corresponding to the antecedent part of rule i, and B_i is the fuzzy set of the consequent part.

Support of a certain rule is equal to the fraction of rules in the RB that exactly composed of the same antecedent and consequent of that particular rule. It is calculated for the ith rule as (Bhargava and Shukla 2016):

$$\text{Sup}(\text{Rule}^i) = \frac{n(A_i\ ,\ B_i)}{m} \tag{2}$$

where $n(A_i\ ,\ B_i)$ returns the number of rules which contain A_i and B_i, simultaneously, and m is the number of available rules in the RB.

Confidence of a certain rule is the proportion of rules that contain both antecedent and consequent parts of that particular rule, to those which only have its antecedent part. It is computed for the ith rule as (Bhargava and Shukla 2016):

$$\text{Conf}(\text{Rule}^i) = \frac{n(A_i\ ,\ B_i)}{n(A_i)} \tag{3}$$

where $n(A_i)$ is the number of rules which contain all A_i^k in their antecedent part.

Support and Confidence are often used to eliminate uninteresting rules in the rule mining algorithms (Bhargava and Shukla 2016). Generally, the strength of each rule is measured by its Support and its Confidence. Indeed, rules with very low Support are not frequent and may hardly occur for a few data samples. On the other hand, a low Confidence rule has a low probability for its occurrence among similar rules (rules whose antecedents are the same as antecedent part of that rule). These rules are not strong and they do not have an effective role in the inferring process, so if a minimum threshold is defined for Support and Confidence, these weak rules are detected and can be removed from the rules set (Batbarai and Naidu 2014).

This common strategy is usually applied in the rule mining algorithms. Indeed, the rules set is evaluated based on some minimum thresholds, and then those rules that failed to reach the minimum thresholds are eliminated from the rules set. The minimum Support and Confidence thresholds are called MinSupp and MinConf, respectively. So, in the mentioned strategy all rules require satisfying MinSupp and MinConf and the rules set is refined through the two steps as follows (Batbarai and Naidu 2014):

1. Find all frequent rules that satisfy MinSupp.
2. Extract all high Confidence rules that satisfy MinConf among the frequent rules that have been found in step 1.

The main difficulty of this strategy is specifying the MinSupp and MinConf measures. If an appropriate threshold is not set, some problems arise; e.g., if the minimum threshold is set too high, some interesting rules may be missed, or if it is set too low, a lot of unnecessary rules may remain. On the other hand, by changing the training data, the minimum thresholds should be updated and adapted with the new data, while finding a user-specified minimum threshold for each dataset is time-consuming, it should be found by trial and error or it needs an expert knowledge. Anyway, setting of these thresholds manually, is a hard and risky mission, and a simpler and more accurate alternative method is required. In this regard, a new stronger rule pruning method is proposed in Sect. 3.2.3.

2.4 A brief overview of PSO

PSO (Du and Swamy 2016) is a parallel search algorithm which is inspired by the swarm behaviors. PSO algorithm tries to find the best solution by starting from some initial solutions and optimizing them continuously. This algorithm produces a population of candidate solutions which are called particles; particles are abstract entities that used to simulate solutions; they can move iteratively and randomly through a multi-dimensional search space; they try to improve their positions as the information changes (He et al. 2016).

Unlike genetic algorithm, PSO has no operators such as crossover and mutation. As Fig. 2 shows, it is initialized with

Fig. 2 Pseudo-code of PSO algorithm (Du and Swamy 2016)

```
For each particle
{ Initialize particle }

Do until maximum iterations or minimum error criteria
{
        For each particle
        {   Calculate fitness value
            If the fitness value is better than Pbest
                {   Set Pbest = current fitness value   }
            If Pbest is better than Gbest
                {   Set Gbest = Pbest   }
        }
        For each particle
        {   Calculate particle velocity
            Update particle position using the current particle solution and velocity
        }
}
```

a population of random particles. These particles are evaluated using a fitness function. Each particle has a velocity, which manages its movement. Indeed, each particle iteratively changes its position in the search space according to two tips: the best-known position found so far by itself called Pbest, and the best-known position found so far by the entire swarm called Gbest. In this way, each particle can move toward the best current solution until the desired convergence criterion is met.

As these statements, each PSO algorithm consists of three steps, namely generating particles' positions and velocities, updating particles' velocities, and updating particles' positions. The pseudo-code of standard PSO algorithm is presented in Fig. 2. More details about update formulas and circumstances of PSO are available in Du and Swamy (2016).

Due to the general advantages[1] of PSO algorithm (Cheng and Jin 2015; Visalakshi and Sivanandam 2009; Oliveira and Schirru 2009), i.e., simplicity, fast convergence, easy implementation, less operators, better computational efficiency, few parameters to adjust, it is employed for our optimization tasks in this paper. However, it is not an obligation and the proposed model can simply adapt with the other optimization algorithms.

3 Description of the proposed method

This section describes details of the three stages of EEFR-R, namely preprocessing, model generation, and post-processing. Figure 3 illustrates the general scheme of EEFR-R. The preprocessing stage prepares the input data through the discretization and data partitioning tasks; it generates a set of CPs using Fayyad and Irani's discretization method. The second stage is the main stage of EEFR-R; it constructs the DB and the RB of a Mamdani FRBS through the three tasks of MFs definition, Rule generation, and Rule pruning. Wang and Mendel's method (WM's) and PSO algorithm assist in performing the rule generation and rule pruning, respectively. Finally, in the post-processing stage, MFs are tuned and rules' weights are adjusted using another PSO algorithm. In what follows, we describe in detail the three stages of EEFR-R.

3.1 Preprocessing

When there is no expert knowledge about MFs (e.g., number of MFs of each variable, or support of each MF), a discretization method is employed to compensate this deficiency; indeed, discretization methods generate a set of CPs which are utilized to define MFs and determine fuzzy partitions.

Regarding the result of Zeinalkhani and Eftekhari (2014), which has recommended Fayyad and Irani's method as one of the most efficient discretization methods in classification tasks, this method is applied in this study to choose CPs and partition data domains. This algorithm is for classification tasks, and it needs a nominal attribute as class label; so, a K-means clustering is performed before it to determine the required nominal outputs. Then, all input and output variables are discretized as described in Sect. 2.2, and a set of CPs is obtained for each variable; the minimum and maximum values of each variable are also added to these sets as the first and last CPs, respectively; i.e., if for variable k with the domain of $[\, l_k\ ,\ u_k\,]$, n CPs are selected, a set S, composed of $n+2$ CPs, is built for it as:

$$S = \left\{ \mathrm{CP}_1^k = l_k,\ \mathrm{CP}_2^k,\ \ldots,\ \mathrm{CP}_{n+2}^k = u_k \right\} \quad (4)$$

Each pair of successive CPs makes one partition; thus, as Fig. 4 illustrates, the domain of variable k is divided into $n+1$

[1] These advantages are general and may be variant depending on the application.

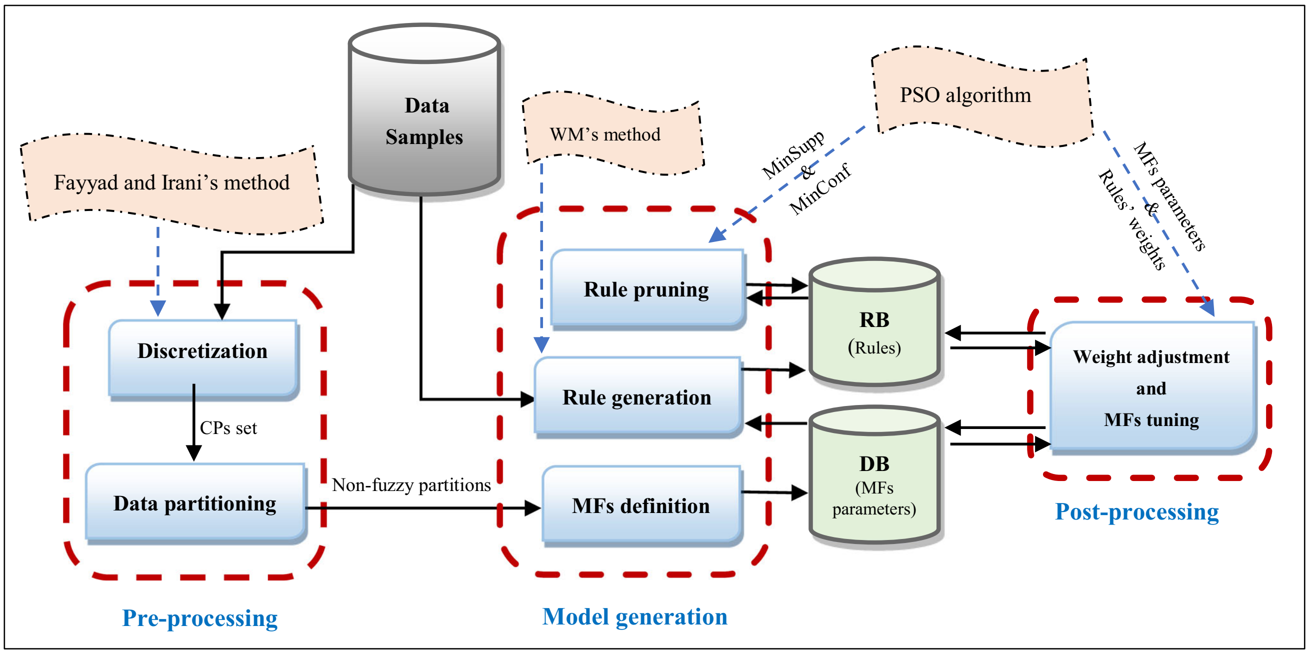


Fig. 3 General scheme of EEFR-R; blue rectangles show the task(s) of each stage, solid black arrows indicate flow of input and output for each task, and dashed blue arrows represent the utilized methods (or algorithms) of each task

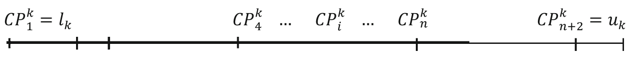


Fig. 4 Partitioning of variable k, using the discretization method

partitions by using the set S. These partitions are employed for definition of MFs in the next section.

3.2 Model generation

To construct a Mamdani FRBS, all of its components including fuzzification, KB, inference engine, and defuzzification should be defined. Among these components, KB, comprised of DB and RB, has a critical role in the fuzzification and inference processes. Hence, in this section, the focus is on the mechanisms of DB and RB generation, and the other components are similar to a typical Mamdani FRBS ("Appendix A"). Since the DB includes fuzzy set definitions and MFs parameters, in the first following subsection, the method of MFs definition is described in details. On the other hand, the RB is composed of a set of If-Then rules which are employed by the inference engine to perform the reasoning operations. In this study, it is proposed to construct the RB via a two-step process: rule generation and rule pruning; these steps are presented to extract all possible rules and to prune additional rules, respectively, and they are demonstrated in the second and third following subsection.

3.2.1 MFs definition

Definition of MFs using discretization methods consists of two steps; In the first step, non-fuzzy partitions are determined using a discretization method, and then in the second step, for each of these partitions an MF is defined (Zeinalkhani and Eftekhari 2014). The operation related to the first step was carried out in the preprocessing stage (Sect. 3.1), and the non-fuzzy partitions corresponding to each variable obtained. In this section, the focus is on transforming non-fuzzy partitions into fuzzy partitions by defining MFs.

MFs are used to map crisp values into fuzzy numbers. For each element X belonging to the fuzzy set A, a degree of membership between 0 and 1 is assigned; it is denoted by $\mu_A(X)$. μ_A is a mathematical function defined using triangular, trapezoidal, Gaussian, or other types of MFs. Among these types of MF, Gaussian MF, due to its advantages in predictive models, is often used for modeling regression problems (Tay and Lim 2011). Gaussian MF has two parameters, c and σ, i.e., it is defined as:

$$\mu_A(X;\, c, \sigma) = \exp\left(-\frac{1}{2}\left(\frac{X-c}{\sigma}\right)^2\right) \tag{5}$$

c is the mean value which represents the center of MF, and it has the curve peak. σ is the standard deviation that controls the curve width.

Four different methods have been proposed in Zeinalkhani and Eftekhari (2014) to define MFs using the generated non-fuzzy partitions. In these methods, different measures are extracted from CPs and partitions, and then based on these measures, MFs are defined; measures such as partition width, standard deviation, neighbor partition coverage rate, and partition coverage rate. In this stage, the method of definition based on partition width is adapted to design MFs. This method utilizes the distance between two CPs to compute the parameters of MFs, and it does not consider how examples are distributed inside each partition.

After the preprocessing tasks of Sect. 3.1, $n+2$ CPs and accordingly $n+1$ partitions were obtained for a typical variable k. In this step, one Gaussian MF is considered for each partition; so, $n+1$ Gaussian MFs must be defined for variable k. Consider the ith MF of variable k is denoted by MF_i^k, it is designed based on partition i which itself has been built using two successive CPs CP_i^k and CP_{i+1}^k. Moreover, as Eq. (5), the two parameters c and σ have to be set for each Gaussian MF, therefore, to define MF_i^k, the values of these two parameters should be determined using the information of CP_i^k and CP_{i+1}^k, i.e., they are specified as follows:

$$\begin{cases} c_i^k = \dfrac{(\mathrm{CP}_{i+1}^k + \mathrm{CP}_i^k)}{2} \\ \sigma_i^k = \dfrac{(\mathrm{CP}_{i+1}^k - \mathrm{CP}_i^k)}{2\sqrt{\ln 4}} \end{cases} \tag{6}$$

This design is based on these principles: the center of each Gaussian MF is placed at the center of each partition, the membership degrees of the two CPs are equal to 0.5, and each two adjacent MFs intersect at one common CP which is the connection point of their respective partitions. Unlike (Zeinalkhani and Eftekhari 2014), the leftmost and the rightmost MFs do not differ from the middle ones. Figure 5 shows an example of MFs definition for a variable in the domain [0.001, 1.0]; a set composed of four CPs as {0.001, 0.1295, 0.5195, 1.0} has been chosen for this variable; CPs have been marked with black solid circles in this figure.

3.2.2 Rule generation

The first step of constructing the RB is rule generation. In this study, the mechanism of rule learning of WM's (Wang and Mendel 1992), with some modifications, is utilized to generate fuzzy If-Then rules. WM's approach is a data-driven method that extracts If-Then rules from data samples. It is based on uniform fuzzy partitioning and performs in five steps. The method of rule generation of this study differs from the WM's one in the fuzzy partitioning mechanism (step 0), and in the determination of the importance degree for each

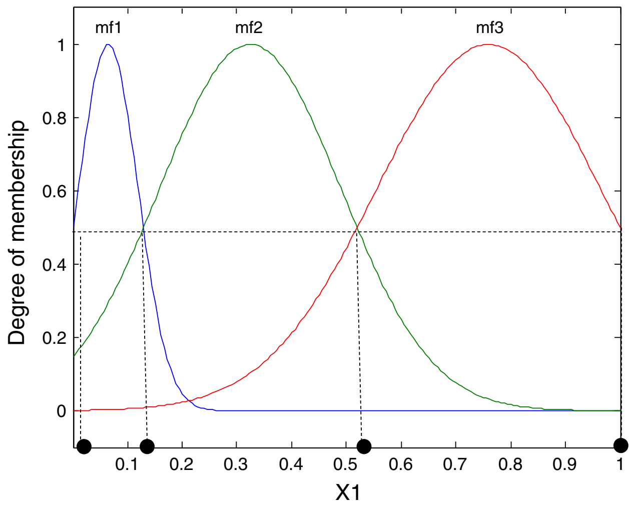


Fig. 5 An example of MFs definition

rule (step 2), i.e., it is carried out through the following four steps as:

Step 0: Discretization, partitioning, and MFs definition for all variables are done according to the principles of Sect. 3.2.1; this step is a prerequisite for the next steps.

Step 1: Extracting one candidate rules from each data sample; suppose that a dataset with m data samples is given, in which each data sample has n input variables and 1 output variable, so the ith row of this dataset is denoted as:

$$\left(X_i^1, \ldots, X_i^n, Y_i\right); \qquad i = 1, \ldots, m \tag{7}$$

According to these assumptions, the structure of each rule has n antecedents, 1 consequent, and totally $n+1$ dimensions. For this data sample, the corresponding linguistic terms of each dimension are determined with the fuzzy set whose output is maximum, same as the original WM's (Wang and Mendel 1992). By repeating this procedure for all m data samples, m candidate rules are obtained, so that each of them may be or may not be a member of the final RB.

Step 2: Assign an importance degree for each candidate rule; the importance degree is a criterion to evaluate strength of each rule in comparison with similar rules in a group of conflicting rules. It is determined by multiplication of Support and Confidence (two rule evaluation metrics introduced in Sect. 2.3), i.e., the importance degree, ID, is assigned for rule i as:

$$\mathrm{ID}(\mathrm{Rule}^i) = \mathrm{Sup}(\mathrm{Rule}^i) * \mathrm{Conf}(\mathrm{Rule}^i) \tag{8}$$

Fig. 6 The proposed rule pruning algorithm; $\mathrm{Sup}(\mathrm{Rule}^i)$ and $\mathrm{Conf}(\mathrm{Rule}^i)$ are calculated as Eqs. (2) and (3), respectively

```
Round 1:   1 − $MinSupp = PSO()$
           2 − $If\ Sup(Rule^i) < MinSupp \Rightarrow RB = RB - \{Rule^i\}$   ; $\forall i = 1\ to\ |RB|$

Round 2:   1 − $MinConf = PSO()$
           2 − $If\ Conf(Rule^i) < MinConf \Rightarrow RB = RB - \{Rule^i\}$   ; $\forall i = 1\ to\ |RB|$
```

where $\mathrm{Sup}(\mathrm{Rule}^i)$ is the Support of rule i as Eq. (2), and $\mathrm{Conf}(\mathrm{Rule}^i)$ is its Confidence as Eq. (3).

Step 3: Classify conflicting rules; rules with exactly the same antecedents but different consequent conflict with each other. Such these rules are classified into one group, and among them, the most appropriate one should be selected.

Step 4: Choose the final fuzzy If-Then rules; from each group, that rule which has the highest importance degree is chosen as a final member of the RB. Regarding the contribution of utilizing the measures of Support and Confidence to define the rules' importance degrees, the strongest rules will be chosen based on their reliability as well as their frequency. In other words, if a rule has the highest degree of frequency (Support) and reliability (Confidence), its importance degree will be the highest and it will be chosen from its group. In this regard, more qualified rules are generated in the first step of constructing the RB.

3.2.3 Rule pruning

In this section, a new rule pruning method is presented to modify the rules set by removing weak rules. Up to this step, a set of If-Then rules with no conflict has been obtained. However, this set is not optimal yet, the number of rules is high, and the set may contain plenty of redundant and not effective rules; so, a modification process is needed to refine this set.

In Sect. 2.3, a common strategy to eliminate additional rules using MinSupp and MinConf was described. The drawbacks of that strategy were also stated. In this section, a new rule pruning method is presented based on that strategy and it is proposed to specify the minimum thresholds automatically and regarding the training data instead of setting fixed values for all situations. For this purpose, the mentioned strategy is combined with an optimization algorithm.

Figure 6 shows the proposed rule pruning algorithm. It is comprised of two rounds. In the first round, the RB is scanned to find the most frequent rules, while in the second round, it is searched to find the most reliable rules. As illustrated in Fig. 6, each round has two steps; at first, a PSO algorithm is run to find the most appropriate minimum threshold, and afterward, the RB is scrutinized and those rules which do not satisfy the minimum thresholds, are eliminated from it.

In this method, finding the appropriate values for MinSupp and MinConf is delegated to the PSO algorithm. It carries out this mission by considering two important perspectives of the rules set, namely cardinality[2] and arising error. Indeed, the RB must be composed of those rules that optimize these two criteria simultaneously as much as possible. To achieve this goal, two new criteria, called Reduction and Increase, are introduced; they are related to the cardinality of the RB and the system error, respectively.

Definitions

- Reduction is defined to control the number of rules in the RB. It is equal to the percentages of diminution in the number of rules after a pruning process, i.e., the Reduction, R, is defined as:

$$R = (1 - \#\mathrm{NR}_{\mathrm{after}}/\#\mathrm{NR}_{\mathrm{before}}) \times 100 \tag{9}$$

  where $\#\mathrm{NR}_{\mathrm{before}}$ and $\#\mathrm{NR}_{\mathrm{after}}$ are the cardinality of the RB before and after the pruning process, respectively.
- Increase is defined to control the system error. After a pruning process, due to eliminating some rules, it is anticipated that the system error rises. Increase criterion is considered to measure this error increment, and it is equal to the percentages of error increment after applying a rule pruning process. Therefore the Increase, I, is defined as:

$$I = (E_{\mathrm{after}}/E_{\mathrm{before}} - 1) \times 100 \tag{10}$$

  where E_{before} and E_{after} are the system errors before and after the rule pruning process, respectively. Mean square error is used to measure the system error in each situation, and it is computed as:

[2] In mathematics, the number of elements of a set is called the cardinality of that set.

$$E = \left(\frac{1}{2 \times |D|}\right) \times \sum_{i=1}^{|D|} \left(F(\overrightarrow{X_i}) - Y_i\right)^2 \tag{11}$$

where $|D|$ is the number of data samples in the training dataset D, $\overrightarrow{X_i}$ is the ith training input vector, $F(\overrightarrow{X_i})$ is the estimated output for this sample using the FRBS, and Y_i is the ith target value.

Fitness function

It is clear that a configuration of the RB, which provides maximum Reduction and minimum Increase simultaneously, is desirable. It means that the maximum number of weak rules should be eliminated, so that the minimum cost be imposed on the system error. PSO algorithm has undertaken this goal. It tries different values of MinSupp and MinConf and evaluates different states of pruning until the best one is found. For this meaning, a fitness function using Reduction and Increase is made for PSO algorithm. Due to the goal of pruning process (maximum Reduction and minimum Increase), and given that PSO algorithm minimizes its fitness function, a fractional function in which Increase is in the numerator and Reduction is in the denominator, is considered as the fitness function of PSO algorithm, i.e., it is formulated as follows:

$$\text{fitness} = \frac{a \times I^k + b}{c \times R^p + d} \tag{12}$$

where I and R are normalized values of Reduction and Increase, respectively. a, b, c, d, k, and p are parameters of the model; they are used to adjust the model to a specified training data. a, b, c, and d ($a \neq 0, c \neq 0$) are coefficients that utilized to make a nonlinear function of Reduction and Increase. They can be usually set to $a = c = 1$ and $b = d = 0$. The most important parameters in this definition are k and p which provides different trade-offs between Reduction and Increase. p is called Reduction impact factor, and it determines the degree of influence of Reduction in the pruning process. Similarly, k is called Increase impact factor, and it specifies the importance of Increase in the pruning process.

Different pruning modes

Depending on the values assigned to the factors k and p, three modes of pruning are provided as follows:

(1) If $k > p \geq 1$, less pruning occurs. In this mode, Increase has a more effective role in the pruning operation rather than Reduction; the system error is in the best situation; and the cardinality of the RB is greater than the other modes.

(2) If $1 \leq k < p$, more pruning occurs. This time, Reduction influences the pruning process more than Increase. In this mode, although the number of rules is less than all other modes, the error increment might be slightly higher.

(3) If $k = p = 1$, a moderate state is obtained. In this mode, the roles of Reduction and Increase in the pruning operation are the same so that the cardinality of the RB and the error increment are in a moderate situation.

Table 1 Different modes of rule pruning

Mode	Parameters	More effective aspect	Result
1	$k > p \geq 1$	Increase	Less pruning
2	$p > k \geq 1$	Reduction	More pruning
3	$k = p$	Both	Moderate

These different modes are also summarized in Table 1. Depending on the application that this model will be applied in, and given the degree of importance of each aspect, a decision maker can set the parameters of the fitness function. According to our experiments, the third mode provides acceptable results in the most applications.

Several remarkable points

In this part, we should mention several remarkable points about the proposed rule pruning method:

- In the employed PSO algorithm, each particle has a two-dimensional position, it is coded as $P = P_{\text{sup}} + P_{\text{conf}}$, where P_{sup} and P_{conf} are related to the MinSupp and MinConf parameters, respectively. These parameters take values in their respective variation intervals. This simple coding scheme provides a significant reduction in the number of required parameters. Indeed, the proposed rule pruning method uses only two parameters for each number of initial rules, while the other rule selection methods (Gacto et al. 2014; Alcalá et al. 2011) have applied one parameter per rule. This reduction is remarkable, especially in the large-scale datasets, where the number of initial rules is probably significantly high. Reducing the length of particle's position and consequently making the search space smaller, improve the efficiency of evolutionary algorithms and lessen the costs of memory and time (Xue et al. 2016); this matter is properly achieved by the proposed rule pruning method, so that our experiments revealed that the employed PSO algorithm converges at most in five iterations, and it effects on the running times that shown in Sect. 4.5.

- In the second mode, more pruning may be caused further increase in the system error; however, this is not a critical issue because a tuning process which compensates this error increasing, is ahead.
- In the fitness function, If p set to 0, the pruning operation will only be performed by considering the system error, and if k set to 0, just the cardinality of the RB will be taken into account; however, neither of these two conditions is recommended for normal applications.
- The parameters a, b, c, and d ($a \neq 0, c \neq 0$) are the coefficients that utilized to make a nonlinear function of Reduction and Increase. They initially and usually set as $a = c = 1$ and $b = d = 0$. However, these parameters are as a safety valve for the fitness function, they can be adjusted more precisely in the second round. Indeed, if manipulating the values of k and p (while a, b, c, and d have their default values) cannot make a capable fitness function, these four parameters are employed to adjust the fitness function more exactly.
- During our experiments, some negative values were observed for the Increase criterion. It means that in some applications with this rule pruning method not only there is not any error rising, but also pruning significantly reduces the system error. It occurs due to the presence of some destructive rules in the RB; these rules are those which lonely and for a few numbers of data samples work well, but they are weak in cooperation with the others; the presence of such these rules make the rules set to weaken, and consequently, the system error grows. The negative Increase values show that the proposed rule pruning method is able to discover these rules, so that when these rules are eliminated from the RB, the system error goes down and the negative values are reported for the Increase criterion.

3.3 Post-processing

Some of the most important factors that influence the performance of an FRBS include the structure of RB, the size of RB, the rules' weight, and the structure of DB. Therefore, to improve the performance of EEFR-R, after generating the initial model, some modifications are organized according to these factors. The first two factors, which are related to the RB, were improved in the rule pruning process, Sect. 3.2.3. In this stage, the final modifications are done, i.e., the structure of DB and the weights of rules are improved by means of another PSO algorithm.

The main constituents of DB are MFs which are specified by their parameters. These parameters should be optimized to find an efficient DB. The mechanism of MFs definition in Sect. 3.2.1 is just a strategy for the initial generation of MFs. It determines the parameters of MFs only by considering the information of the CPs and regardless of the training data. This is a deficiency that should be compensated through the optimization of MFs.

The second goal of this stage is the proper adjustment of the rules' weights. According to our empirical results, when a rule's weight is initially set with the Confidence of that rule, fewer errors will occur. This is not happened by chance, and it was predictable; because the weight of each rule determines the strength of influence of that rule, and this is the same as the reliability of that rule which was previously (in Sect. 2.3) introduced as Confidence criterion. However, this initial setting for the rules' weights is not enough, and again the training data are not considered. Thus, in this regard, an optimization process is also required.

A PSO algorithm is employed to perform the aforementioned optimization tasks. To codify the particle's position in this algorithm, a double-coding scheme is considered as:

$$P = P_{\mathrm{MF}} + P_W \tag{13}$$

The first part of this scheme is P_{MF} (Vaneshani and Jazayeri-Rad 2011); it is used to codify parameters of all available MFs in the DB, i.e., it is encoded as:

$$\begin{aligned} &P_{\mathrm{MF}} = P^1, P^2, \ldots, P^{n+1} \\ &P^i = c_1^i, \sigma_1^i, \ldots, c_{NMF(i)}^i, \sigma_{NMF(i)}^i; \qquad i = 1, \ldots, n+1 \end{aligned} \tag{14}$$

in which as Eq. (7) the system has $n + 1$ variables, n inputs and 1 output. If we suppose that $NMF(i)$ MFs have been defined for the ith variable, P^i encodes all parameters of all MFs related to this variable. Since the type of MFs are Gaussian, and given that each Gaussian MF has 2 parameters, the length of P_{MF} is calculated as:

$$|P_{\mathrm{MF}}| = \sum_{i=1}^{n+1} 2 \times NMF(i) \tag{15}$$

By adjusting parameter c in this scheme, each MF, as Fig. 7a shows, can move its peak between its two constructor CPs. It is a kind of lateral tuning (Alcalá et al. 2007) which shifts the position of an MF to its environment until the best possible position is found. On the other hand, as Fig. 7b, by adjusting parameter σ, each MF can change its width up to twice of its initial value. In fact, the tuning of MFs is a combination of these two changes, simultaneously. An example of MFs before and after tuning is shown in Fig. 7c and d, respectively.

The second part of the particle's scheme is P_W, and it is used to codify the rules' weights. Assume that the number of final rules (after rule pruning) is equal to m, so the weights of these m rules are encoded in P_W as:

$$P_W = (W_1, W_2, \ldots, W_m) \tag{16}$$

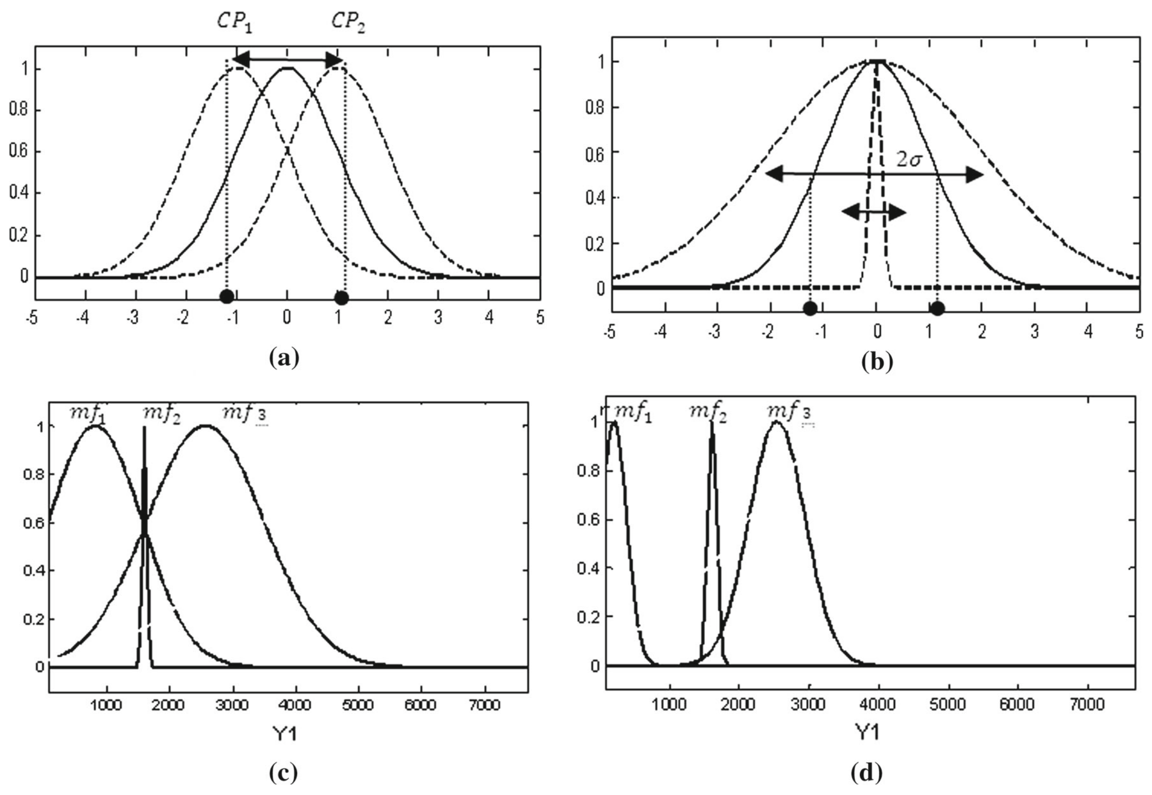


Fig. 7 **a** Adjusting the center of MF, **b** adjusting the width of MF, **c** MFs before the tuning, **d** MFs after the tuning

where W_i is the weight of rule ith that can obtain an optimal value between 0 and 1. After the codification, an initial set of particles is randomly generated. Then, the particles are evaluated using the fitness function until the stopping conditions are met. The fitness function is equal to the system error as Eq. (11). The mechanisms of updating for the positions and the velocities are similar to what is done in the standard PSO (Du and Swamy 2016).

Since in this stage, the process of optimization deals with the data samples, the applied evolutionary algorithm may be computational in the case of large-scale datasets, which have a high number of data samples; therefore, an efficient strategy should be considered to tackle this problem. For this purpose, the mechanism of fast error estimation, proposed in Alcalá et al. (2011), has been incorporated into the PSO algorithm of this stage. It uses a small percentage of data samples for initial evaluation of a new solution; if, given the initial evaluation, that solution is not dominated, the evaluation must be repeated using the whole data samples; but if that solution is dominated, it is discarded, and the process continues with the next solution. With this mechanism, the whole set of data samples is used only to evaluate the solutions which seem desirable. This method improves the evolutionary algorithms in terms of computations and running times. More details about the error estimation mechanism are available in Gacto et al. (2014), Alcalá et al. (2011).

4 Experimental results

In this section, the experiments, which have been carried out to evaluate the effectiveness of EEFR-R, are detailed. In what follows, firstly the datasets and the parameters of the experiments are described. Then, different stages of the proposed model are individually evaluated, and in the fourth subsection, the overall performance of EEFR-R is compared with the three state-of-the-art approaches. The statistical tests are also conducted to further analyzing of these methods. At the end of this section, the average running times of EEFR-R are reported.

4.1 Datasets, parameters and evaluation criteria

To evaluate EEFR-R, 19 real-world regression datasets have been used; they have different number of variables, ranging from 2 to 40, and different number of data samples ranges from 337 to 14998. All datasets have been chosen from the KEEL dataset repository (Alcalá-Fdez et al. 2011b). Table 2 shows the main characteristics of these datasets. It shows the number of variables and the number of data samples for each dataset. All experiments were implemented using MATLAB tools and keel software (Alcalá-Fdez et al. 2011b). STAC platform (Rodríguez-Fdez et al. 2015) was used to do the

Table 2 Properties of the used datasets for the experimental study

Datasets	Abbr.	# of variables	# of data samples
Electrical Length 1	Ele-1	2	495
Plastic Strength	PLA	2	1650
Quake	Quake	3	2178
Electrical Length 2	Ele-2	4	1056
Friedman	Freidman	5	1200
AutoMPG6	autoMPG6	5	398
Delta Ailerons	Del-Ail	5	7129
Daily Electricity Energy	Dee	6	365
AutoMPG8	autoMPG8	7	398
Weather Izmir	WIZ	9	1461
Stock prices	stock	9	950
Forest Fires	FOR	12	517
Mortgage	MOR	15	1049
Treasury	TRE	15	1049
Baseball	BAS	16	337
Computer Activity	CA	21	8192
Pole Telecommunications	POLE	26	14998
Pumadyn	PUM	32	8192
Ailerons	AIL	40	13750

statistical tests. The mechanism of 10-fold cross-validation was utilized to generate the training and test data. Moreover, each experiment was performed three times for each fold, and the average of 30 runs were reported as final result. The parameters of EEFR-R have been set as follows; they are general and fixed for all 19 datasets in all experiments, i.e.,

- The properties of Mamdani FIS which is generated using EEFR-R are specified in Table 3.
- The parameters of PSO algorithm, employed in the rule pruning method of Sect. 3.2.3 and in the post-processing operation of Sect. 3.3, are adjusted in Table 4.

In the case of evaluation criteria, different FRBSs are usually compared in terms of their efficiency in the two notable aspects, namely accuracy and complexity. So, two criteria, related to each of these aspects, have been considered for the evaluation of this stage. The error of estimation, as Eq. (11), is considered as the accuracy measure; it is denoted by Tra. for the training data and by Tst. for the test data in the following tables. On the other hand, to evaluate the complexity or clearly the interpretability of the FRBSs, there are no explicit and fixed criteria in the literature. Different measures have been used in different applications (Alonso et al. 2015). In this model, given the contribution of pruning of additional rules, the complexity is considered at the level of RB, and the most well-known measure of this level, namely the cardinality of the RB, is utilized, i.e., it is denoted by $\#R$ in the following tables.

Table 3 FIS structure

FIS properties	Method
Type	Mamdani
AND operator	Min
OR operator	Max
Implication method	Min
Aggregation method	Max
Defuzzification	Centroid (center of gravity)

Table 4 PSO algorithm parameters

Parameter	Value
C1	1.2
C2	0.8
Inertia W factor	1
Number of population	100
Iterations (in the tuning stage)	100
Iterations (in the pruning stage)	10

4.2 The role of rule pruning process

Two criteria, namely Reduction, as Eq. (9), and Increase, as Eq. (10), were introduced to propose the rule pruning method. The same criteria were also considered for evaluation of it. The focus of pruning method is to reduce the number of

Table 5 Results of different rule pruning modes

Datasets	Less pruning		Moderate		More pruning	
	$k = 2, p = 1$		$k = 1, p = 1$		$k = 1, p = 2$	
	R (%)	I (%)	R (%)	I (%)	R (%)	I (%)
Ele-1	88.25	− 13.17	88.91	− 12.67	89.75	− 10.88
PLA	63.28	15.89	77.00	46.43	82.14	70.30
Quake	90.14	− 18.12	94.39	− 15.34	95.09	− 14.80
Ele-2	84.22	47.33	88.69	51.95	92.65	60.48
Freidman	77.29	17.36	83.40	29.03	90.44	49.38
autoMPG6	86.41	− 1.94	87.28	− 0.03	90.78	7.14
Del-Ail	76.06	− 35.31	88.84	− 27.13	91.28	− 19.41
Dee	83.47	27.6	91.15	36.50	94.65	49.38
autoMPG8	88.47	− 7.7	96.60	− 3.38	96.88	− 1.96
WIZ	90.17	− 5.67	93.69	− 1.53	97.69	1.07E−04
stock	67.82	15.7	81.45	38.34	91.08	79.31
FOR	88.05	− 0.17	90.05	− 0.0072	94.31	2.93
MOR	91.53	− 0.0083	93.54	0.0094	97.54	0.14
TRE	87.47	− 0.0091	90.17	− 0.0079	91.73	1.06
BAS	76.09	− 2.50E−15	88.11	3.04E−15	92.74	7.71E−15
CA	80.23	1.07E−06	95.04	1.97E−06	97.02	2.13E−06
POLE	73.10	− 4.517	84.38	− 1.69	87.90	15.23
PUM	50.15	1.12E−15	52.50	1.17E−15	61.111	1.30E−14
AIL	87.97	7.83	91.97	14.59	94.39	18.59

rules (Reduction) as much as possible, so that the least error increment (Increase) is arisen.

Table 5 shows the three modes of rule pruning method in the three columns, namely less pruning mode (first), moderate mode (second), and more pruning mode (third), from left to right, respectively. For each dataset, the values of Reduction and Increase of all modes have been indicated. As expected, for all datasets, Reduction has its minimum value in the first mode and its maximum value in the third mode. Furthermore, in the first mode, due to the least pruning, the minimum Increase is occurred, and vice versa, in the third mode, due to the most pruning, the maximum Increase is observed. In other words, in Table 5, by moving from the less pruning mode toward the more pruning mode, for all datasets, Reduction values grow and accordingly Increase values also grow.

From another perspective, as previously stated, the presence of some rules in the RB may be destructive, so that eliminating of them will result in error decrement instead of error increment; these cases are revealed by negative Increase values. The proposed rule pruning algorithm, due to its fitness function, is able to discover these situations. According to the results of Table 5, in the first mode 11 cases, in the second mode 9 cases, and in the third mode 4 cases of such situations have occurred.

Generally, the experiments that performed in the moderate mode, provide the best balances between Reduction and Increase. In this mode, in 9 datasets, there is no error increment, and in 4 datasets (MOR, BAS, CA, and PUM), the increment is negligible; so, it can be concluded that the pruning process does not have an adverse effect on the system error, even it has a positive influence in the most cases. In the 6 remaining datasets, the error increments are slightly high; however, it does not worry us, and they are compensated at the tuning stage.

Lastly, we mention that the experiments of this stage were performed with the default values k and p; in this way, the fractional fitness function is formed by the minimal difference between its numerator and denominator. It is clear that by adjusting the larger values for k and p so that the difference between numerator and denominator becomes more, the functionality of each mode will be more confirmed, and the results will be more robust.

4.3 The role of MFs and weight tuning

In this section, the effectiveness of the post-processing stage is evaluated. Table 6 shows the results of this evaluation. This table has three main columns. The first two columns are related to two situations before and after the tuning process, and the third column is for Reduct measure. In each situation, the errors of the training and the test data have been indicated. As can be seen, for all datasets, the values of Tra. and Tst. significantly reduce after the tuning process. However, to

Table 6 Results of MFs and weight tuning, the errors in this table should be multiplied by 10^5, 10^{-8}, 10^5, 10^{-4}, 10^{-8} in the case of Ele-1, Del-Ail, BAS, PUM, AIL, respectively

Datasets	Before tuning		After tuning		Reduct (%)	
	Tra.	Tst.	Tra.	Tst.	Tra.	Tst.
Ele-1	3.61	3.66	2.34	2.00	35.06	45.36
PLA	2.39	2.36	1.34	1.07	43.90	54.36
Quake	0.0198	0.0198	0.0179	0.0171	**9.97**	**14.02**
Ele-2	1.93E+04	1.89E+04	7046	7422	63.49	60.73
Freidman	4.30	4.53	2.87	2.15	33.10	52.51
autoMPG6	8.69	8.98	4.29	5.19	50.60	42.19
Del-Ail	3.83	3.88	3.13	3.17	18.27	18.29
Dee	0.127	0.145	0.075	0.072	40.69	50.39
autoMPG8	8.11	8.52	4.76	5.19	41.32	39.06
WIZ	4.63	4.54	0.53	0.94	88.44	79.14
stock	2.55	2.85	0.63	0.73	75.31	74.39
FOR	2097.3	2079.7	2005.5	2023.3	4.377	2.711
MOR	2.80	2.76	0.11	0.12	**95.75**	**95.65**
TRE	1.099	1.096	0.228	0.255	79.25	76.73
BAS	3.42	3.49	2.19	2.78	35.96	20.34
CA	7.64	7.54	4.17	4.19	45.41	44.43
POLE	124.70	129.89	93.20	98.70	25.26	24.01
PUM	3.79	4.00	0.41	0.45	89.11	88.55
AIL	3.50	4.80	2.92	3.62	16.78	24.56
Average					**46.95**	**47.76**

analyze more, Reduct measure is defined. It is used to find out the percentage of error reduction after applying the tuning process, i.e., it is calculated as:

$$\text{Reduct} = (1 - E_{\text{after}}/E_{\text{before}}) \times 100 \tag{17}$$

where E_{before} and E_{after} are the system errors (as Eq. (11)) before and after the tuning process, respectively. Reduct values are different in every case of Table 6; the minimum values are for dataset Quake (9.970 for the Tra. and 14.02 for the Tst.), while the maximum values are for dataset MOR (95.75 for the Tra. and 95.65 for the Tst.); these values are marked in bold in Table 6.

As the last row of Table 6 shows, the average values of Reduct are 46.95 and 47.76 for the train and test errors, respectively. Given these values, it can be deduced that the post-processing stage generally halves the errors, and this is an effective complementary task to improve the accuracy of the model.

4.4 Comparing the overall performance of methods

In this section, the overall performance of EEFR-R is evaluated in comparison with the three state-of-the-art regression models. Therefore, in the next two subsections, at first, the three selected models are introduced, and then, results of comparisons are presented.

4.4.1 Selected methods from the literature for comparisons

Three different methods have been considered to evaluate the performance of EEFR-R. These methods include one classical data-driven method and two evolutionary fuzzy systems. A brief review of these models is as follows:

- Wang and Mendel's method (Wang and Mendel 1992) is an ad-hoc data-driven method which learns fuzzy rules from data example through a five-step algorithm. Since this method utilizes grid partitioning to define MFs, and given that it chooses the best rules based on a powerless importance degree, it involves some drawbacks, e.g., an enormous number of rules or ignoring cooperation of the rules. However, due to its simplicity and quickness, it has been employed in many researches and comparisons. In our experiments, it has been implemented with five labels and denoted by WM (5).
- $\text{FS}_{\text{MOGFS}}{}^{\text{e}} + \text{TUN}^{\text{e}}$(Alcalá et al. 2011) is a fast and scalable multi-objective genetic fuzzy system for linguistic fuzzy modeling in high-dimensional regression problems. It proposes to learn all components of the KB of a Mamdani FIS, simultaneously in a common process. Moreover, lateral tuning of MFs and rule selection are done to further refinement of the model.
- FRULER (Rodríguez-Fdez et al. 2016), fuzzy rule learning through evolution for regression, is a new genetic fuzzy system to learn a TSK FIS automatically. It has

Table 7 Average results of different methods, the test errors in this table should be multiplied by 10^5, 10^{-8}, 10^5, 10^{-4}, 10^{-8} in the case of Ele-1, Del-Ail, BAS, PUM, AIL, respectively

Datasets	EEFR-R		$FS_{MOGFS}{}^e + TUN^e$		FRULER		WM(5)	
	$\#R$	Tst.	$\#R$	Tst.	$\#R$	Tst.	$\#R$	Tst.
Ele-1	**3.5**	2.000	8.1	**1.954**	4.1	2.012	17	5.055
PLA	13.3	**1.078**	18.6	1.194	**1.4**	1.219	15	6.1414
Quake	3.7	**0.0171**	**3.2**	0.0178	7.8	0.0181	54	0 .052
Ele-2	10.1	7422	8	10548	**4.3**	**6729**	65	56359
Freidman	**6.5**	2.151	22	3.138	8	**0.731**	839	4.9016
autoMPG6	**10.8**	4.196	20	4.562	13.7	**3.727**	115	6.096
Del-Ail	**1.7**	2.17	15	2	2.5	**1.458**	221	5.89
Dee	11.1	**0.072**	18.3	0.093	**7.9**	0.080	193	0.2165
autoMPG8	**9.3**	5.195	23	4.474	12.7	**4.084**	161	7.195
WIZ	**6.3**	0.947	10	1.011	8.9	**0.663**	399	5.961
stock	**13.3**	0.732	23	0.912	42.4	**0.353**	265	2.9101
FOR	**3.3**	**2023**	10	2628	5.6	2214	375	34235
MOR	**6.1**	0.012	7	0.019	7.9	**0.007**	199	0.134
TRE	**3.4**	0.035	9	0.044	4.5	**0.027**	196	0.405
BAS	**4.3**	2.78	17	**2.61**	6.2	3.05	253	6.19
CA	**3.2**	**4.198**	14	5.216	7.1	4.634	1539	12.44
POLE	16.3	**98.7**	**13.1**	102.81	40.8	110.898	3461	473.48
PUM	**4.1**	0.45	17.6	**0.29**	7.8	0.36	7372	4.29
AIL	**7.3**	1.62	15	2	8.5	**1.40**	6581	2.99

been organized in the three stages. At first, a new instance selection process performs to prepare data. Next, a multi-granularity fuzzy discretization obtains non-uniform fuzzy partitions of the input variables, and finally, a genetic algorithm learns the fuzzy rules based on the elastic net regularization mechanism.

4.4.2 Results of comparisons

Table 7 shows the average results of EEFR-R and the three selected methods. In this table, each column has been dedicated to one method. Two measures of $\#R$ and Tst. have been taken into account to compare these methods. The best values of $\#R$ and Tst. for each row have been marked in bold.

As Table 7 shows, about $\#R$, EEFR-R has the lowest values in the majority of the datasets, 14 out of 19 cases; after that, FRULER has 3 cases, and $FS_{MOGFS}{}^e + TUN^e$has 2 cases of the best values. By these results, it seems that EEFR-R operates better than the other methods in terms of complexity reduction in the level of RB. However, to further investigate this initial guess, the statistical tests are carried out in the next section.

About Tst. values, Table 7 shows EEFR-R has obtained the lowest errors in 6 cases, FRULER in 10 cases, and $FS_{MOGFS}{}^e + TUN^e$in 3 cases. So, FRULER has generally generated more accurate models. However, it seems that EEFR-R does not have a significant difference with FRULER. The final conclusion about these results needs more investigation and comparison that are done in the next section.

Table 8 Results of Friedman's test on $\#R$. ($\alpha = 0.1$)

Algorithm	Rank
EEFR-R	1.31
FRULER	2.05
$FS_{MOGFS}{}^e + TUN^e$	2.68
WM (5)	3.94
p-value = 1E−5	H_0 is rejected

4.4.3 Statistical nonparametric tests

In what follows, the statistical tests are done in order to analyze the results of Table 7 more. According to the recommendation of the assistant of STAC platform (Rodríguez-Fdez et al. 2015), Friedman's test (Friedman 1937) is the best fitted test for our data. It is a nonparametric statistical test which ranks the models based on a specific measure. Friedman's test was performed for both considered measures of Table 7 ($\#R$ and Tst.).

Table 8 shows the results of Friedman's test for $\#R$; P value of the test has been indicated in the last row. Given the Rank values, EEFR-R has the top ranking level; FRULER, $FS_{MOGFS}{}^e + TUN^e$, and WM (5) are at the next levels, respectively.

Table 9 Result of Holm's test for $\#R$, EEFR-R as the control method and $\alpha = 0.1$

Comparison	Statistic	Adjusted p-value	Hypothesis
EEFR-R versus FRULER	1.75	0.078	Rejected
EEFR-R versus $FS_{MOGFS}{}^{e} + TUN^{e}$	3.26	0.002	Rejected
EEFR-R versus WM (5)	6.28	1E−5	Rejected

In the next step, to find out which of the methods have significant differences, a Holm's post hoc test was conducted. This test evaluates a null hypothesis (H_0) and compares a control method with the remaining methods in pairs. Table 9 shows the results of this test with EEFR-R as the control method and significance level $\alpha = 0.1$. Since the adjusted p values of all comparisons are less than α, Holm's test rejects the null hypotheses of all comparisons. It means that in the measure of $\#R$, the methods FRULER, $FS_{MOGFS}{}^{e} + TUN^{e}$, and WM (5) are not statistically equivalent with EEFR-R, and there are significant differences between EEFR-R and the others. Thus, due to the top ranking of EEFR-R in the complexity criterion, and given the significant differences of EEFR-R with the other approaches, it is concluded that EEFR-R and the proposed rule pruning method have been successful in reducing the complexity of the model.

In order to analyze accuracy of the methods, Friedman's test was also performed for the Tst. values of Table 7. Table 10 shows the results and the p-value of this test. Give the rank values, FRULER is in the first ranking place, and EEFR-R, with a little difference, is in the second level. In this regard, since FRULER employs a TSK FIS and EEFR-R utilizes a Mamdani FIS, and the TSK systems are usually more accurate than the Mamdani ones, the ranking results of accuracy were predictable and also acceptable.

Similarly, Holm's post hoc test was performed again to find out whether the difference between EEFR-R and the first method, FRULER, is significant or not. This time, FRULER was the control method. As Table 11 shows, the hypothesis of equality of FRULER and EEFR-R is accepted. It indicates that although EEFR-R is placed at the second level of ranking, there is no significant difference between it and the first method, and EEFR-R can work as accurate as FRULER.

In addition, to compare the accuracy of EEFR-R with its next ranked method in Table 10, a Wilcoxon's test (Wilcoxon 1945) was performed. It is also a statistical test to do the individual pairwise comparison. Table 12 shows the result of Wilcoxon's test for EEFR-R versus $FS_{MOGFS}{}^{e} + TUN^{e}$with $\alpha = 0.1$. According to the adjusted p-value and the values of R^{+} and R^{-} in Table 12, the null hypothesis is rejected in favor of EEFR-R. It means that EEFR-R is more successful than $FS_{MOGFS}{}^{e} + TUN^{e}$in the Tst. measure.

With respect to these analyses for the complexity and accuracy, it is inferred that EEFR-R has achieved the most accurate solution with the least complexity. In order to demonstrate the functionality of EEFR-R more clearly, an illustrative example of it is provided in "Appendix B."

Table 10 Results of Friedman's test on Tst. ($\alpha = 0.1$)

Algorithm	Rank
FRULER	1.73
EEFR-R	1.84
$FS_{MOGFS}{}^{e} + TUN^{e}$	2.42
WM (5)	4.00
p-value = 1E−5	H_0 is rejected

Table 11 Result of Holm's test on Tst., FRULER as the control method and $\alpha = 0.1$

Comparison	Statistic	Adjusted p-value	Hypothesis
FRULER versus EEFR-R	0.25	0.80	Accepted

Table 12 Results of Wilcoxon's test on Tst., EEFR-R versus $FS_{MOGFS}{}^{e} + TUN^{e}$and $\alpha = 0.1$

Comparison	R^{+}	R^{-}	Adjusted p-value	Hypothesis
EEFR-R versus $FS_{MOGFS}{}^{e} + TUN^{e}$	144	46	0.048	Rejected

4.5 Time evaluation

Table 13 shows the average running time for the different stages of EEFR-R. The time of run of EEFR-R has been also calculated for each dataset. These times were obtained using a single thread of an Intel Xeon Processor E5-2650L (20M Cache, 1.80 GHz). As can be seen, the first two stages of EEFR-R, which perform the fundamental operations of this model including model generation and rule pruning, are not time-consuming; their time ranges from 5 seconds to 21 min for Ele-1 problem and the most complex problem AIL, respectively. The main portion of the total time of EEFR-R is related to the post-processing stage which focuses on the tuning and error decreasing. This time ranges from 43 seconds to 1 h and 4 min in the cases of Ele-1 and AIL, respectively. Indeed, except for the most complex datasets (CA, POLE, PUM, AIL), EEFR-R could obtain the model in less than 10 min. The total times of the complex datasets, given their number of variables and/or samples, are also very

Table 13 Average running time for different stages of EEFR-R (HH:MM:SS)

Datasets	Ele-1	PLA	Quake	Ele-2	Freidman	autoMPG6	Del-Ail	Dee	autoMPG8	WIZ
Discretization	00:00:01	00:00:01	00:00:02	00:00:01	00:00:01	00:00:01	00:00:05	00:00:01	00:00:01	00:00:02
Model_gen and pruning	00:00:04	00:00:10	00:00:08	00:00:26	00:00:20	00:00:03	00:00:20	00:00:06	00:00:19	00:00:23
Post-processing	00:00:43	00:03:59	00:01:56	00:09:25	00:03:51	00:01:55	00:02:53	00:01:16	00:01:33	00:04:38
Total time	00:00:48	00:04:10	00:02:09	00:09:52	00:04:12	00:01:59	00:03:18	00:01:23	00:01:53	00:05:03
Datasets	stock	FOR	MOR	TRE	BAS	CA	POLE	PUM	AIL	
Discretization	00:00:02	00:00:01	00:00:02	0:00:02	00:00:01	00:00: 11	00:00:08	00:00: 11	00:00: 14	
Model_gen and pruning	00:00:16	00:00:06	00:00:12	00:00:45	00:00:19	00:05:17	00:17:01	00:15:27	00:20:11	
Post-processing	00:02:33	00:00:51	00:04:50	00:6:34	00:02:39	00:07:34	00:33:51	00:10:29	01:03:51	
Total time	00:02:51	00:00:58	00:05:04	00:7:21	00:02:59	00:13:02	00:50:00	00:26:07	01:24:16	

good (less than an hour and a half). These times are obtained, due to the complexity reduction that EEFR-R provides well, especially in the complex datasets.

5 Conclusion

This paper proposed EEFR-R, a novel rule extraction method integrated into an evolutionary fuzzy system for regression problems. EEFR-R formed three stages to learn and modify a Mamdani fuzzy inference system based on Wang and Mendel's and the association rule mining concepts. Indeed, the DB and the RB of a Mamdani FRBS were initially generated during the first two stages, and then, they were refined through some modifications in the last stage. Furthermore, a new rule pruning method was proposed in order to eliminate weak rules and refine the RB based on the preferences of the decision makers. Nineteen real-world regression datasets were used to evaluate the performance of EEFR-R. Results of evaluations and statistical tests revealed that EEFR-R obtained the simplest model with a high degree of accuracy.

Nowadays, large-scale and big datasets have made several challenges for machine learning algorithms. As future works, the proposed method can be integrated into the multi-objective evolutionary fuzzy systems. It can be also adapted to handle the large-scale and big data challenges, particularly. For instance, feature selection and training set selection algorithms can be added. Furthermore, some parallel and distributed computing frameworks such as Map-Reduce can be used for utilizing the memory and CPU resources optimally.

Acknowledgements This study was not funded at all.

Compliance with ethical standards

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

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

Appendix A: overview of Mamdani FRBSs

The standard architecture of a fuzzy rule-based system consists of four main modules as below (Riza et al. 2015):
Fuzzification—The fuzzification module converts the crisp input values into linguistic ones. Linguistic values refer to ordinary concepts like high, medium, low, etc., which are used in our conversation to declare measures. MFs are defined based on these linguistic variables to map input values into fuzzy concepts.

Knowledge Base (KB)—KB is one of the essential components of FRBS. It comprises two fundamental parts including Data Base (DB) and Rule Base (RB). The DB includes fuzzy set definitions and the MF's parameters while the RB contains a set of linguistic fuzzy If-Then rules. Fuzzy rules are applied in the decision-making process. They are written in the following format: **If** x is A, **Then** y is B where the If part is called antecedent and the Then part is called consequent of the rule. This rule is described as: **If** the antecedent conditions are satisfied, Then the consequent can be inferred.
Inference Engine—The inference engine is where FRBS performs the process of reasoning; it uses fuzzy If-Then rules and input data to make the inference operation.
Defuzzification module—In defuzzification procedure, the acquired output is transformed from a fuzzy value into a crisp one.

Among the different fuzzy inference systems (Timothy 2010), two types of Mamdani FIS (Mamdani 1977) and Takagi–Sugeno–Kang (TSK) FIS (Takagi and Sugeno 1985) have been widely used in the regression applications. The most important differences between these FISs are related to the consequent part of the rules, and accordingly, their aggregation and defuzzification methods are different. In Mamdani fuzzy inference systems, both antecedent and consequent parts of a rule are determined by fuzzy sets, while in TSK systems only the antecedent part is represented by fuzzy sets and the consequent part is computed through a weighted linear function or a constant value. Mamdani fuzzy systems are usually applied to get more interpretable models, whereas TSK systems focus on the accuracy and precision (Blej and Azizi 2016).

The general form of a fuzzy If-Then rule in Mamdani system is like this:

$$\begin{aligned}\text{Rule}^i :&\textbf{If } X^1 \textit{ is } A_i^1 \textit{ and } X^2 \textit{ is } A_i^2 \textit{ and } \ldots \textit{ and } X^n \textit{ is } A_i^n \\ &\textbf{Then } Y \textit{ is } B_i;\ i = 1,\ \ldots,\ p \end{aligned} \tag{18}$$

where X^k is the kth input variable, n is the number of input variables, Y is the output variable, p is the number of all rules, and A_i^j and B_i are the linguistic variables defined by MFs in the DB. The main advantage of this composition of inputs and output is its close relationship with the way of human thinking.

After generating or learning the constituents of DB and RB, the process of inferring in Mamdani system is done through the following five steps:

(1) Fuzzification of the input values using the defined MFs.
(2) Applying the fuzzy operator (AND or OR) to the antecedents in order to combine the fuzzified inputs and obtain the rule strength (h_i). h_i is also called firing strength of the ith rule. It measures the degree of matching the input vector ($X^1, X^2, \ldots, X^n$) to the ith rule.

$$h_i = T(A_i^1(X^1), A_i^2(X^2), \ldots, A_i^n(X^n)) \tag{19}$$

where A_i^k is the fuzzy value of the kth input variable (X^k) and T is the T-norm conjunctive operator. Mamdani specifically recommended the use of the minimum T-norm (Mamdani 1977).
(3) Indicating the final consequent for each rule; it is achieved by combining the computed rule strength and the linguistic fuzzy term related to the output of a certain rule as:

$$B_i'(Y) = T(h_i, B_i(Y)) \tag{20}$$

B_i' is the conclusive consequent of the ith rule, B_i is the fuzzy value of output variable Y and T is the T-norm conjunctive operator. Mamdani has also recommended the use of the minimum T-norm in this case (Mamdani 1977).
(4) Aggregation of all rule's consequent; this is usually done by using the fuzzy OR operator as:

$$O(Y) = \bigcup_{i=1}^{p} B_i'(Y) \tag{21}$$

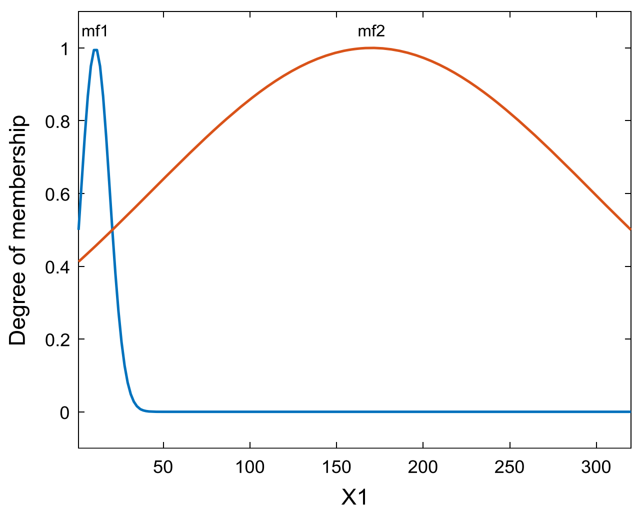
where $O(Y)$ is the result MF.
(5) Defuzzification of the result; the output MF is not enough in many applications and the crisp value is needed. Some methods like centroid, weighted average, mean-max membership, etc., have been suggested computing the crisp value of the result. The most common method is centroid which uses the center of gravity of the fuzzy sets in which the crisp output y is calculated as:

$$y = \int O(Y)Y\mathrm{d}y \Big/ \int O(Y)\mathrm{d}y \tag{22}$$

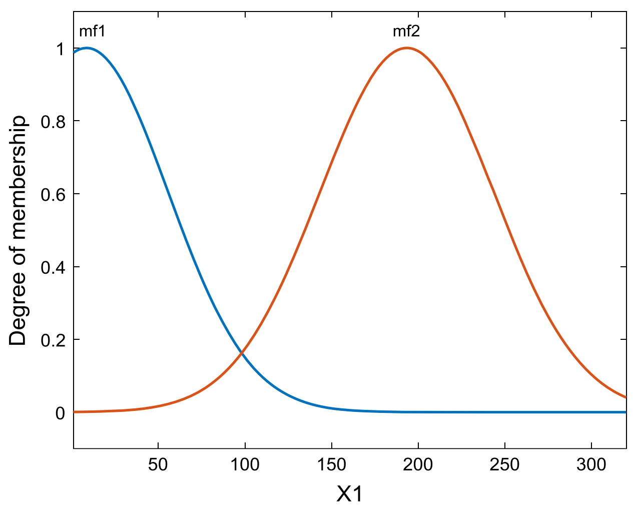
Due to the capability of Mamdani FIS in generating simple and interpretable models and regarding our objective in reducing the complexity of regression solutions, it is used in this study. Mamdani weakness in getting lower accuracy (in comparison with TSK models), however, has been overcome using optimization methods.

Appendix B: an illustrative example of EEFR-R

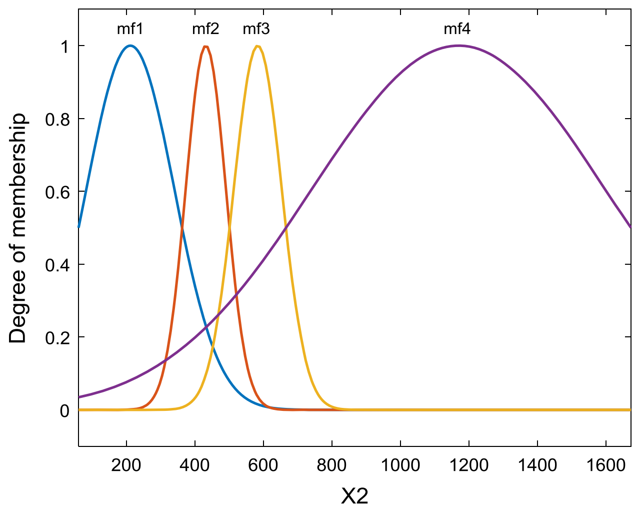
In this section, an illustrative example of EEFR-R is described. For this purpose, the first problem of Table 2, Ele-1, has been taken into account. Ele-1 is a real-world benchmark problem which estimates the length of low-voltage lines in rural towns using some available inputs. This



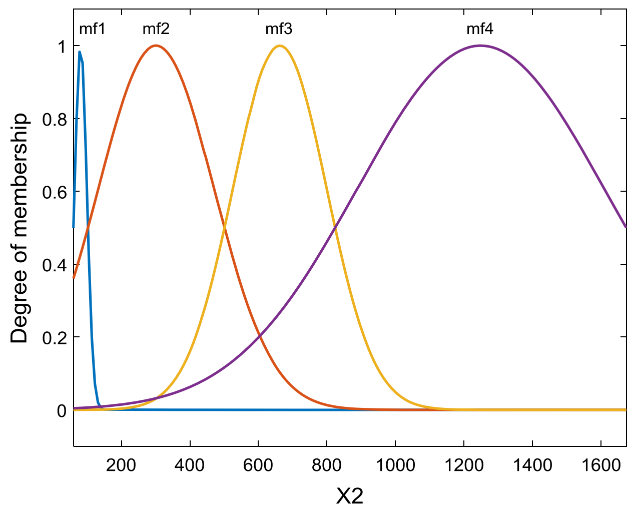
(a) The initial MFs of Inhabitants



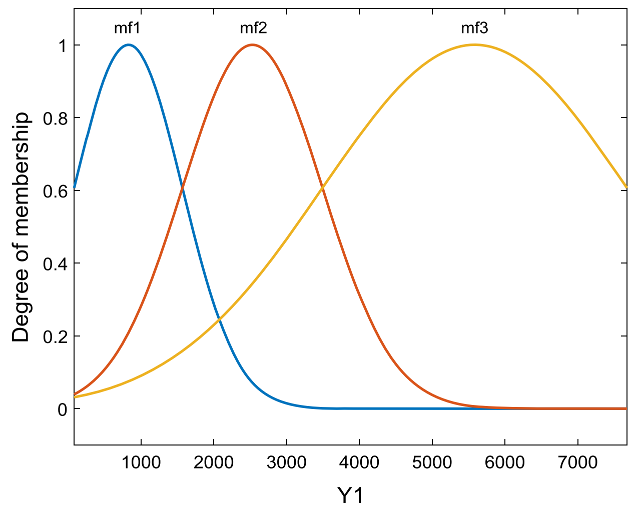
(b) The tuned MFs of Inhabitants



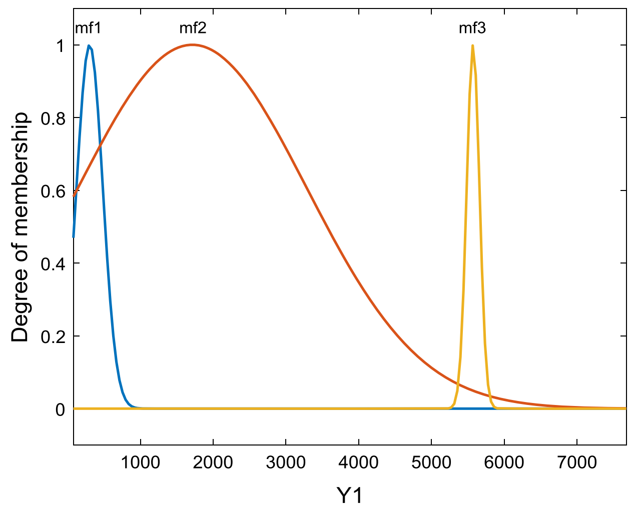
(c) The initial MFs of Distance



(d) The tuned MFs of Distance



(e) The initial MFs of Length



(f) The tuned MFs of Length

Fig. 8 MFs of different variables; left figures are related to the initial MFs, and right figures are the same left MFs which have been tuned

dataset includes 3 attributes, namely Inhabitants, Distance, and Length; Inhabitants and Distance are the system inputs which are used to estimate the output Length (Cordón et al. 1999).

EEFR-R is performed for Ele-1 in the three stages as the descriptions of Sect. 3. In the following, the results of one run of EEFR-R for this dataset are demonstrated. In the preprocessing stage, Sect. 3.1, a set of CPs is generated for each variable as:

$$\begin{aligned} S_{\text{Inhabitants}} &= \{1,\ 20,\ 320\} \\ S_{\text{Distance}} &= \{60,\ 362.5,\ 500.8,\ 665.83,\ 1673.3\} \\ S_{\text{Length}} &= \{80,\ 1568,\ 3492.5,\ 7675\} \end{aligned}$$

These sets are utilized in the model generation stage to define initial MFs in Sect. 3.2.1; the initial MFs of the three variables are shown in the left column of Fig. 8. In the following of the second stage, all rules are extracted from the data samples of the Ele-1 dataset using the rule generation process of Sect. 3.2.2; these rules and their weights are represented in Table 14. After that, the rule pruning method, 3.2.3, is carried out, and among the 7 generated rules, just 3 of them (the highlighted ones in italic) remain in the RB. Table 15 shows the structure of final RB.

Finally, as Sect. 3.3, the post-processing tasks including MFs tuning and rules' weights adjustment are done. The right column of Fig. 8 shows the MFs of the three variables after the tuning. Also, the optimized weights of the final rules are indicated in the weight column of Table 15. The test error of this run of EEFR-R for the Ele-1 dataset was equal to 2.0527E+05.

Table 14 Results of rule generation

#	Inhabitants	Distance	Length	Weight
1	mf1	mf1	mf1	1.00
2	*mf1*	*mf2*	*mf1*	*0.98*
3	mf1	mf3	mf2	0.64
4	mf1	mf4	mf2	0.43
5	*mf2*	*mf2*	*mf1*	*0.71*
6	*mf2*	*mf3*	*mf2*	*0.83*
7	mf2	mf4	mf2	0.55

Table 15 Results of rule pruning and rules' weights adjustment

#	Inhabitants	Distance	Length	Weight
1	*mf1*	*mf2*	*mf1*	1
2	*mf2*	*mf2*	*mf1*	0.79
3	*mf2*	*mf3*	*mf2*	0.60

References

Alcalá R, Alcalá-Fdez J, Herrera F (2007) A proposal for the genetic lateral tuning of linguistic fuzzy systems and its interaction with rule selection. IEEE Trans Fuzzy Syst 15(4):616–635

Alcalá R, Gacto MJ, Herrera F (2011) A fast and scalable multi-objective genetic fuzzy system for linguistic fuzzy modeling in high-dimensional regression problems. IEEE Trans Fuzzy Syst 19(4):666–681

Alcalá-Fdez J, Alcala R, Herrera F (2011a) A fuzzy association rule-based classification model for high-dimensional problems with genetic rule selection and lateral tuning. IEEE Trans Fuzzy Syst 19(5):857–872

Alcalá-Fdez J, Fernández A, Luengo J, Derrac J, García S, Sánchez L, Herrera F (2011b) Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. J Mult-Valued Log Soft Comput 17(2–3):255–287

Alonso JM, Castiello C, Mencar C (2015) Interpretability of fuzzy systems: current research trends and prospects. In: Springer handbook of computational intelligence. Springer, pp 219–237

Antonelli M, Bernardo D, Hagras H, Marcelloni F (2017) Multiobjective evolutionary optimization of type-2 fuzzy rule-based systems for financial data classification. IEEE Trans Fuzzy Syst 25(2):249–264

Batbarai A, Naidu D (2014) Survey for rule pruning in association rule mining for removing redundancy. Int J Innov Res Sci Eng Technol 3(4):11313–11315

Bhargava N, Shukla M (2016) Survey of interestingness measures for association rules mining: data mining, data science for business perspective. Analysis 6(2):2249–9555

Blej M, Azizi M (2016) Comparison of mamdani-type and sugeno-type fuzzy inference systems for fuzzy real time scheduling. Int J Appl Eng Res 11(22):11071–11075

Cheng R, Jin Y (2015) A social learning particle swarm optimization algorithm for scalable optimization. Inf Sci 291:43–60

Cordón O, Herrera F, Sánchez L (1999) Solving electrical distribution problems using hybrid evolutionary data analysis techniques. Appl Intell 10(1):5–24

Dash R, Paramguru RL, Dash R (2011) Comparative analysis of supervised and unsupervised discretization techniques. Int J Adv Sci Technol 2(3):29–37

de Sá CR, Soares C, Knobbe A (2016) Entropy-based discretization methods for ranking data. Inf Sci 329:921–936

Debie E, Shafi K, Merrick K, Lokan C (2014) An online evolutionary rule learning algorithm with incremental attribute discretization. In: IEEE congress on evolutionary computation (CEC), pp 1116–1123

Du K, Swamy M (2016) Particle swarm optimization. Springer, Berlin

Elragal H (2010) Using swarm intelligence for improving accuracy of fuzzy classifiers. Int J Electr Comput Energ Electron Commun Eng 4(8):11–19

Esmin AAA (2007) Generating fuzzy rules from examples using the particle swarm optimization algorithm. In: International conference on hybrid intelligent systems, pp 340–343

Fayyad U, Irani K (1993) Multi-interval discretization of continuous-valued attributes for classification learning. In: Proceedings of the 13th international joint conference on artificial intelligence, pp 1022–1027

Fernandez A, Lopez V, del Jesus MJ, Herrera a (2015) Revisiting evolutionary fuzzy systems: taxonomy, applications, new trends and challenges. Knowl-Based Syst 80:109–121

Friedman M (1937) The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J Am Stat Assoc 32(200):675–701

Gacto MJ, Galende M, Alcalá R, Herrera F (2014) METSK-HD^e: a multi-objective evolutionary algorithm to learn accurate tsk-fuzzy systems in high-dimensional and large-scale regression problems. Inf Sci 276:63–79

Garcia S, Luengo J, Sáez JA, Lopez V, Herrera F (2013) A survey of discretization techniques: taxonomy and empirical analysis in supervised learning. IEEE Trans Knowl Data Eng 25(4):734–750

He Y, Ma WJ, Zhang JP (2016) The parameters selection of pso algorithm influencing on performance of fault diagnosis. MATEC Web Conf 63:02019

Jang J-S (1993) ANFIS: adaptive-network-based fuzzy inference system. IEEE Trans Syst Man Cybern 23(3):665–685

Kato ER, Morandin O, Sgavioli M, Muniz BD (2009) Genetic tuning for improving wang and mendel's fuzzy database. In: IEEE international conference on systems, man and cybernetics, pp 1015–1020

Liu H, Cocea M (2018) Granular computing-based approach of rule learning for binary classification. Granul Comput 4(2):1–9

Mamdani EH (1977) Application of fuzzy logic to approximate reasoning using linguistic systems. IEEE Trans Comput 26(12):1182–1191

Oliveira MVd, Schirru R (2009) Applying particle swarm optimization algorithm for tuning a neuro-fuzzy inference system for sensor monitoring. Prog Nucl Energy 51(1):177–183

Patel M (2013) Various rule pruning techniques and accuracy measures for fuzzy rules. Int J Appl Innov Eng Manag 2(12):175–178

Permana KE, Hashim SZM (2010) Fuzzy membership function generation using particle swarm optimization. Int J Open Probl Compt Math 3(1):27–41

Prasad M, Chou K-P, Saxena A, Kawrtiya OP, Li D-L, Lin C-T (2014) Collaborative fuzzy rule learning for mamdani type fuzzy inference system with mapping of cluster centers. In: IEEE symposium on computational intelligence in control and automation, pp 1–6

Ratner B (2017) Statistical and machine-learning data mining: techniques for better predictive modeling and analysis of big data. Chapman and Hall/CRC, London

Riza LS, Bergmeir CN, Herrera F, Benítez Sánchez JM (2015) FRBS: Fuzzy rule-based systems for classification and regression in R. J Stat Softw 65(6):1–30

Rodríguez-Fdez I, Canosa A, Mucientes M, Bugarín A (2015) Stac: a web platform for the comparison of algorithms using statistical tests. In: IEEE international conference on fuzzy systems (FUZZ-IEEE), pp 1–8

Rodríguez-Fdez I, Mucientes M, Bugarín A (2016) FRULER: fuzzy rule learning through evolution for regression. Inf Sci 354:1–18

Shehzad K (2013) Simple hybrid and incremental postpruning techniques for rule induction. IEEE Trans Knowl Data Eng 25(2):476–480

Shill PC, Akhand M, Murase K (2011) Simultaneous design of membership functions and rule sets for type-2 fuzzy controllers using genetic algorithms. In: International conference on computer and information technology (ICCIT), pp 554–559

Takagi T, Sugeno M (1985) Fuzzy identification of systems and its applications to modeling and control. IEEE Trans Syst Man Cybern 15(1):116–132

Tay KM, Lim CP (2011) Optimization of gaussian fuzzy membership functions and evaluation of the monotonicity property of fuzzy inference systems. In: IEEE international conference on fuzzy systems (FUZZ-IEEE), pp 1219–1224

Timothy J et al (2010) Fuzzy logic with engineering applications. Wiley, Chichester

Vaneshani S, Jazayeri-Rad H (2011) Optimized fuzzy control by particle swarm optimization technique for control of cstr. World Acad Sci Eng Technol 59:686–691

Visalakshi P, Sivanandam S (2009) Dynamic task scheduling with load balancing using hybrid particle swarm optimization. Int J Open Probl Compt Math 2(3):475–488

Wang LX, Mendel JM (1992) Generating fuzzy rules by learning from examples. IEEE Trans Syst Man Cybern 22(6):1414–1427

Wilcoxon F (1945) Individual comparisons by ranking methods. Biom Bull 1(6):80–83

Xue B, Zhang M, Browne WN, Yao X (2016) A survey on evolutionary computation approaches to feature selection. IEEE Trans Evol Comput 20(4):606–626

Zadeh L (1965) Fuzzy sets. Inform Control 8(3):338–353

Zadeh L (1975) The concept of a linguistic variable and its application to approximate reasoning-i. Inf Sci 8(3):199–249

Zanganeh M, Yeganeh-Bakhtiary A, Bakhtyar R (2011) Combined particle swarm optimization and fuzzy inference system model for estimation of current-induced scour beneath marine pipelines. J Hydroinform 13(3):558–573

Zeinalkhani M, Eftekhari M (2014) Fuzzy partitioning of continuous attributes through discretization methods to construct fuzzy decision tree classifiers. Inf Sci 278:715–735