

# FRULER: Fuzzy Rule Learning through Evolution for Regression

I. Rodríguez-Fdez\*, M. Mucientes, A. Bugarín

*Centro Singular de Investigación en Tecnoloxías da Información (CiTIUS), Universidade de Santiago de Compostela, SPAIN*

---

## Abstract

The use of Takagi-Sugeno-Kang (TSK) fuzzy systems in regression problems is widely extended due to the precision of the obtained models. Moreover, the use of simple linear TSK models is usually referred as a good choice in many real problems since it provides a straightforward functional relationship between the output and input variables. In this paper we present FRULER (Fuzzy RULE Learning through Evolution for Regression), a new genetic fuzzy system for automatically learning accurate and simple linguistic TSK fuzzy rule bases for regression problems. FRULER achieves a low complexity of the learned models while keeping a high accuracy, by following three stages: instance selection, multi-granularity fuzzy discretization of the input variables, and evolutionary learning of the rule base combined with Elastic Net regularization to obtain the consequents of the rules. Each of these stages was validated using 28 real-world datasets. FRULER was also experimentally compared with three state of the art genetic fuzzy systems, showing the most accurate and simple models even when compared with approximative approaches.

*Keywords:* Genetic Fuzzy Systems, regression, instances selection, multi-granularity fuzzy discretization

---

## 1. Introduction

Predictive models usually have two complementary requirements: accuracy and interpretability [5, 4]. On the one hand, accuracy indicates the ability of the model to predict values close to the real ones. On the other hand, interpretability refers to the easiness with which people understand the model [5]. A number of predictive models involving fuzzy rule-based systems have aimed to combine the interpretability and expressiveness of the rules with the ability of fuzzy logic for representing uncertainty [8, 9]. In this regard, interpretability for fuzzy systems involves two main issues [5]:

- **Readability:** it is related to the simplicity of the structure of the fuzzy system itself, i.e., the number of variables, linguistic terms per variable, fuzzy rules, premises per rule, etc. It represents the quantitative or objective part of the model interpretability.
- **Comprehensibility:** it is determined by the general semantics of the fuzzy system and the fuzzy inference mechanism. It is associated with the meaning of the fuzzy partitions for the user. It represents the qualitative or subjective part of the interpretability.

In fuzzy systems, the most important aspect to be considered in terms of interpretability is the definition of the fuzzy partition for each variable, also called the data base definition. Two different approaches can be used for this definition: (i) the linguistic approach, in which all rules share the same fuzzy partition for each variable; (ii) the approximative approach, which may have a different definition of the fuzzy labels for each rule in the rule base. The former implies more interpretability through a higher simplicity and comprehensibility, while the latter usually obtains more accurate solutions. However, approximative approaches can lead to complex partitions of the input space, which

---

\*Corresponding author. Tel.: +34 8818 16392.

*Email addresses:* ismael.rodriguez@usc.es (I. Rodríguez-Fdez), manuel.mucientes@usc.es (M. Mucientes), alberto.bugarin.diz@usc.es (A. Bugarín)

makes harder to understand and capture the insights of the relationship between inputs and outputs. Moreover, strong linguistic partitions are labelled as the most interpretable choice in this regard, since they fulfil all the relevant semantic constraints such as distinguishability, coverage, normality, convexity and others [5].

In the case of fuzzy models for regression problems, two different approaches have been proposed in the literature: Mamdani fuzzy systems (where both antecedent and consequent are represented by fuzzy sets) and Takagi-Sugeno-Kang (TSK) fuzzy systems [36, 35], where the antecedents are represented by fuzzy sets, while the consequents are a weighted combination of the input variables. Although Mamdani systems are well-known because of their interpretability, the linear model in TSK rules is also a good choice since it is straightforward to understand the relationship between the output and input variables. This is of particular interest in many fields, such as robotics [32, 29, 26], medical imaging [30], industrial estimation [27] and optimization of processes [38].

One of the most widely used learning algorithms for automatic building of fuzzy rule bases are Genetic Fuzzy Systems (GFSs) [10], i.e., the combination of evolutionary algorithms and fuzzy logic. Evolutionary algorithms are appropriate for learning fuzzy rules due to their flexibility—that allows them to codify any part of the fuzzy rule base system—and due to their capability for managing the balance between accuracy and simplicity of the model. In particular, recent developments using multi-objective evolutionary fuzzy systems can be found in [2, 15, 33, 6], where both Mamdani and TSK systems were proposed to solve large-scale regression problems. Moreover, in [25] an adaptive fuzzy inference system was proposed to cope with high-dimensional problems.

The simplicity of the models obtained by GFSs for regression has been mostly achieved in the literature through keeping the number of rules and/or the number of labels in the rule base as low as possible using multi-objective approaches [12, 20]. More recently, the use of instance selection techniques has received more attention in both classification [16, 14] and regression [31] problems. This approach faces two problems at once: decreases the complexity for large-scale problems and reduces the overfitting, as the rules can be generated with a part of the training data and the error of the rule base can be estimated with another part (or the whole) training set. Moreover, when no expert knowledge is available to determine the fuzzy labels, two different approaches can be applied: uniform discretization combined with lateral displacements [1], or non-uniform discretization [21]. Recently, [13, 17] have shown the application of non-uniform discretization techniques to classification problems.

The use of TSK fuzzy rule bases implies another complexity dimension: the polynomial in the consequent—usually with degree 1 (TSK-1) or 0 (TSK-0). The most widely used approach for learning the coefficients of the polynomial is the least squares method. However, this choice often leads to models that overfit the training data and misbehave in test. This problem can be solved by shrinking (Ridge regularization) or setting some coefficients to zero (Lasso regularization), thus obtaining simpler models. Moreover, a combination of both regularizations, called Elastic Net [40] can be used.

In this paper we present FRULER (Fuzzy RULE Learning through Evolution for Regression), a new GFS algorithm for obtaining accurate and simple linguistic TSK-1 fuzzy rule base models to solve regression problems. The simplicity of the fuzzy system aims to improve the readability of the model—and, therefore, its interpretability—by obtaining linguistic fuzzy partitions with a low number of labels, a low number of rules, and through the regularization of the consequents—which reduces the number of input variables that contribute to the output. The main contributions of this work are: i) a new instance selection method for regression, ii) a novel multi-granularity fuzzy discretization of the input variables, in order to obtain non-uniform fuzzy partitions with different degrees of granularity, iii) an evolutionary algorithm that uses a fast and scalable method with Elastic Net regularization to generate accurate and simple TSK-1 fuzzy rules.

This work is structured as follows. Section 2 defines the TSK model used by FRULER. Section 3 describes the different stages of the GFS: the instance selection method, the discretization approach, and the evolutionary algorithm. Sec. 4 shows the results of the approach in 28 regression problems, and the comparison with other proposals involving statistical tests. Finally, the conclusions are presented in Sec. 5.

## 2. Takagi-Sugeno-Kang fuzzy rule systems

Takagi, Sugeno, and Kang proposed in [35, 36] a fuzzy rule model in which the antecedents are comprised of linguistic variables, as in the case of Mamdani systems [22, 23], but the consequent is represented as a polynomial function of the input variables. These type of rules are called TSK fuzzy rules. The most common function for the

consequent of a TSK rule is a linear combination of the input variables (TSK-1), and its structure is as follows:

$$\begin{aligned} &\text{If } X_1 \text{ is } A_1 \text{ and } X_2 \text{ is } A_2 \text{ and } \dots \text{ and } X_p \text{ is } A_p \text{ then} \\ &Y = \beta_0 + X_1 \cdot \beta_1 + X_2 \cdot \beta_2 + \dots + X_p \cdot \beta_p \end{aligned} \quad (1)$$

where  $X_j$  represents the  $j$ -th input variable,  $p$  the number of input variables,  $A_j$  is the linguistic fuzzy term for  $X_j$ ,  $Y$  is the output variable, and  $\beta_j$  is the coefficient associated with  $X_j$  in the consequent part of the rule.

The matching degree  $h$  between the antecedent of the rule  $r_k$  and the current inputs to the system  $(x_1, x_2, \dots, x_p)$  is calculated as:

$$h_k = T(A_1^k(x_1), A_2^k(x_2), \dots, A_p^k(x_p)) \quad (2)$$

where  $A_j^k$  is the linguistic fuzzy term for the  $j$ -th input variable in the  $k$ -th rule and  $T$  is the t-norm conjunctive operator, usually the minimum function. The final output of a TSK fuzzy rule base system composed by  $m$  TSK fuzzy rules is computed as the average of the individual rule outputs  $Y_k$  weighted by the matching degree:

$$\hat{y} = \frac{\sum_{k=1}^m h_k \cdot Y_k}{\sum_{k=1}^m h_k} \quad (3)$$

Linguistic TSK fuzzy rule systems represent a good trade-off between accuracy and interpretability, since:

- The use of linguistic terms in the antecedent of the rules provides a full description of the input space due to the shared definition of the fuzzy partitions in the data base of the system.
- The linear representation of the output allows the system to obtain accurate solutions using different well-studied statistical methods.
- The consequent of the rules represented by a linear combination of the input variables facilitates the understanding of the relationship between the inputs and the output.

Thus, even if the TSK fuzzy rule systems are less comprehensible in natural language terms than Mamdani approaches, the system can provide useful and understandable information, and is the preferable choice in some domains. In this article we focus on developing simple and accurate TSK fuzzy rule models based on a linguistic representation of the antecedents.

### 3. FRULER description

This section presents the three main components of FRULER: a two-stage preprocessing —formed by the instance selection and multi-granularity fuzzy discretization modules—, and a genetic algorithm, which contains an ad-hoc TSK-1 (first order) rule generation module (Fig. 1). Both preprocessing techniques are executed to improve the simplicity of the fuzzy rule bases obtained by the evolutionary algorithm. On the one hand, instance selection reduces the variance of the models focusing the generated rules on the representative examples. On the other hand, multi-granularity fuzzy discretization decreases the complexity of the fuzzy partitions, thus making unnecessary to establish an upper bound in the number of rules in the evolutionary stage.

The evolutionary learning process obtains a definition of the data base for each knowledge system. Then a novel ad-hoc TSK-1 rule generation module calculates the antecedents and consequents of each rule using only the representative examples. Finally, each knowledge base is evaluated using the full training dataset.

#### 3.1. Instance Selection for Regression

The instance selection method for regression is an improvement of the CCISR (Class Conditional Instance Selection for Regression) algorithm [31], which is an adaptation for regression of the instance selection method for classification CCIS (Class Conditional Instance Selection) [24]. The main differences between the FRULER instance selection process and CCISR are:

- The error measure is based on the 1 – *nearest neighbor* (1NN) approach for regression, thus reducing the complexity of the calculations compared with CCISR, which uses an ad-hoc fuzzy system to evaluate the instances.

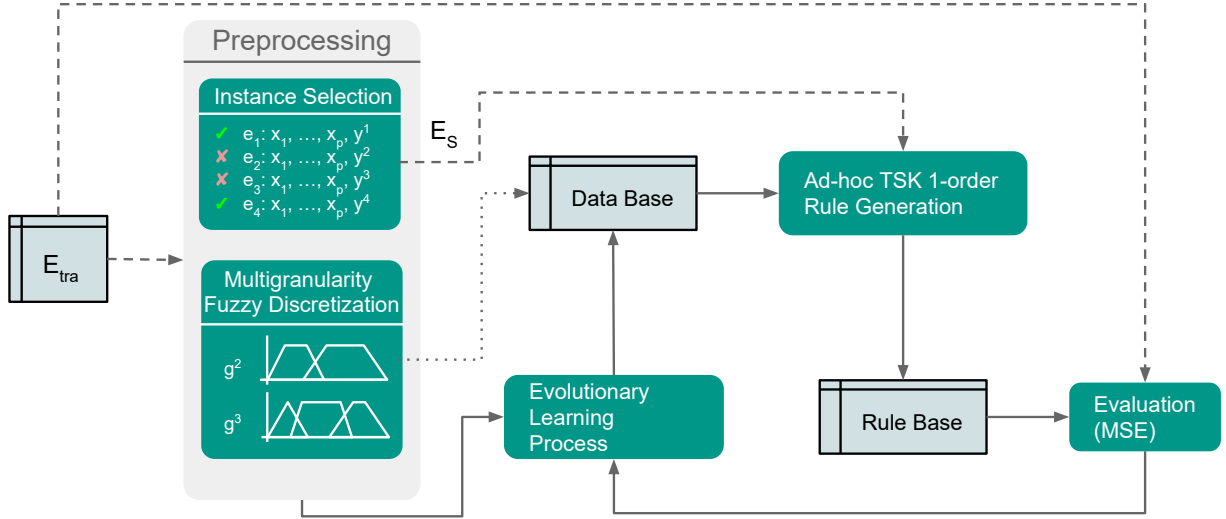


Figure 1: FRULER architecture showing each of the three separated stages. Dashed lines indicate flow of datasets, dotted lines multigranularity information and solid lines represent process flow.

- The stopping criteria is more flexible, allowing more iterations without improvement until the termination of the process.
- The size of the initial set of selected examples is also different due to the previous improvements.

The instance selection process is based on a relation called class conditional nearest neighbor (*ccnn*) [24], defined on pairs of points from a labeled training set as follows: for a given class  $c$ , *ccnn* associates to instance  $a$  its *nearest neighbor* computed among only those instances (excluded  $a$ ) in class  $c$ . This relation, therefore, describes proximity information conditioned to a class label.

In regression problems, the outputs are real values instead of labels and, therefore, they must be discretized in order to use the *ccnn* relation. Traditionally, an unsupervised discretization process needs the definition of either the number of intervals or their shape [11]. In FRULER, the shape of the intervals takes the output density into account, i.e., the intervals are selected such that they represent dense clusters. In other words, the split points between intervals are selected in the zones where the output density is locally minimum.

We use Kernel Density Estimation (KDE) with a gaussian kernel in order to estimate the probability density function of the output variable ( $y$ ) in a non-parametric way. In order to select the appropriate kernel bandwidth, Scott's rule is applied. [34]. Once the probability density function is obtained, the local minimum determines the split points, and, therefore, which labels/classes are used for the *ccnn* relation. Thus, each instance is associated with one of the labels obtained by this process, and the instance selection method can follow the CCIS procedure.

Two different graphs can be constructed using this relation, as proposed in CCIS:

- Within-class directed graph ( $G_{wc}$ ): consists of a graph where each instance has an edge pointing to the nearest instance of the same class.
- Between-class directed graph ( $G_{bc}$ ): is a graph where each instance has an edge pointing to the nearest instance of any different class.

These graphs are used to define an instance scoring function by means of a directed information-theoretic measure (the K-divergence) applied to the in-degree distributions of these graphs. The Score scoring function is defined as:

$$Score(e^i) = p_w^i \cdot \log\left(\frac{p_w^i}{(p_w^i + p_b^i)/2}\right) - p_b^i \cdot \log\left(\frac{p_b^i}{(p_w^i + p_b^i)/2}\right) \quad (4)$$

where  $e^i$  is the example considered,  $p_w^i$  is the in-degree of  $e^i$  in  $G_{wc}$  divided by the total in-degree of  $G_{wc}$ , and  $p_b^i$  is the inner degree of  $e^i$  in  $G_{bc}$  divided by the total in-degree of  $G_{bc}$ . This scoring function is used to develop an effective large margin instance selection method, called Class Conditional selection (Fig. 2).

The instance selection algorithm starts from a set of training examples  $E$ . The method uses the *leave-one-out* mean squared error (MSE) with  $1NN$  (this error is called  $\epsilon$ ) in order to estimate the information loss. Thus, although the scoring function and the graphs are based on the labels obtained by KDE, the error measure is based on the original regression problem.

---

```

1:  $E = \{e^1, \dots, e^n\}$  sorted in decreasing order of Score (Eq. 4)
2:  $S = \{e^1, \dots, e^{k_0}\}$ 
3:  $it_{wi} = 0$ 
4: repeat
5:    $Temp = S \cup \{e^l\}$ 
6:   if  $\epsilon^{Temp} < \epsilon^S$  then
7:      $it_{wi} = 0$ 
8:      $S_{best} = Temp$ 
9:   else
10:     $it_{wi} = it_{wi} + 1$ 
11:     $S = Temp$ 
12: until  $E = S \vee it_{wi} > \sqrt{|E|/|S|}$ 
13: return  $S_{best}$ 

```

---

Figure 2: Pseudocode of Class Conditional selection [24].

First, an initial core of instances from  $E$  is selected, sorted by Score (Fig. 2, lines 1-2). The size of this initial set is:

$$k_0 = \max\left(c, \left\lceil \frac{\epsilon^E \cdot |E|}{\max(y) - \min(y)} \right\rceil\right) \quad (5)$$

where  $c$  is the number of classes obtained from KDE and  $\epsilon^E$  is the error using the set of examples in  $E$ . This choice is motivated by (i) at least one example for each class is considered, and (ii) the error in the second part can be interpreted as the miss-classification probability divided by the range of the output  $\max(y) - \min(y)$ . Thus, the second part indicates that at least the miss-classified examples must be selected in order to be correctly classified. After this, the instance selection method iteratively selects instances and adds them to the set  $S$  (lines 4-12), choosing in the first place those with the highest score. The process terminates when all the examples of  $E$  are in  $S$  or when  $it_{wi}$  —the number of consecutive iterations for which the empirical error ( $\epsilon^S$ ) increases— is greater than  $\sqrt{|E|/|S|}$  (line 12). This threshold allows more iterations without improvement at the beginning of the selection process, when the error is more sensitive, and stops earlier when the number of selected instances is high.

In order to further improve the number of selected instances, CCIS uses the Thin-out post-processing (Fig. 3). This algorithm selects points close to the decision boundary of the  $1NN$  rule. This is achieved by selecting instances having positive in-degree in the between-class graph set  $S$  ( $G_{bc}^S$ ) and storing them in  $S_f$ . Then, an iterative process is performed as follows: points having positive in-degree in the  $G_{bc}^{S_1}$  are added to  $S_f$  in case they were not “isolated” in the previous iteration, that is, if their in-degree was non-null (line 6). This iterative process terminates when the empirical error increases (line 7).

### 3.2. Multi-granularity Fuzzy Discretization for Regression

The definition of the fuzzy partition of each input variable is a critical step in the design of TSK fuzzy rule bases. When no knowledge is available, the set of fuzzy labels for a variable is automatically obtained through fuzzy discretization. Moreover, if the number of labels is unknown, then a multi-granularity approach may be used. In a multi-granularity proposal, each regarded granularity has a different fuzzy partition. Specifically, a granularity  $g_{var}^i$  divides the variable  $var$  in  $i$  fuzzy labels, i.e.,  $g_{var}^i = \{A_{var}^{i,1}, \dots, A_{var}^{i,i}\}$ .

---

```

1:  $S_f = \{e^l \in S \text{ with in-degree in } G_{bc}^S > 0\}$ 
2:  $S_{prev} = S$ 
3:  $S_1 = S \setminus S_f$ 
4:  $go\_on = true$ 
5: while  $go\_on$  do
6:    $S_t = \{e \in S_1 \text{ with in-degree in } G_{bc}^{S_1} > 0 \text{ and with in-degree in } G_{bc}^{S_{prev}} > 0\}$ 
7:    $go\_on = e^{S_f \cup S_t} < e^{S_f}$ 
8:   if  $go\_on$  then
9:      $S_f = S_f \cup S_t$ 
10:     $S_{prev} = S_1$ 
11:     $S_1 = S \setminus S_f$ 
12: return  $S_f$ 

```

---

Figure 3: Pseudocode of Thin-out selection [24].

The generation of the fuzzy linguistic labels can be divided into two stages. First, the variable must be discretized to obtain a set of split points  $C^g$  for each granularity  $g$ . Then, given the split points, the fuzzy labels can be defined for each granularity. In a top-down approach, the split points are searched iteratively, i.e., only a new split point is added at each step, obtaining two new intervals. Therefore, the approach proposed in this work aims to preserve interpretability between contiguous granularities: adding a new label to the previous granularity and modifying the flanks of the adjacent labels (Fig. 4). In regression problems (TSK-1 in our case), the discretization process must search for the split point that minimizes the error when a linear model is applied to each of the resulting intervals.

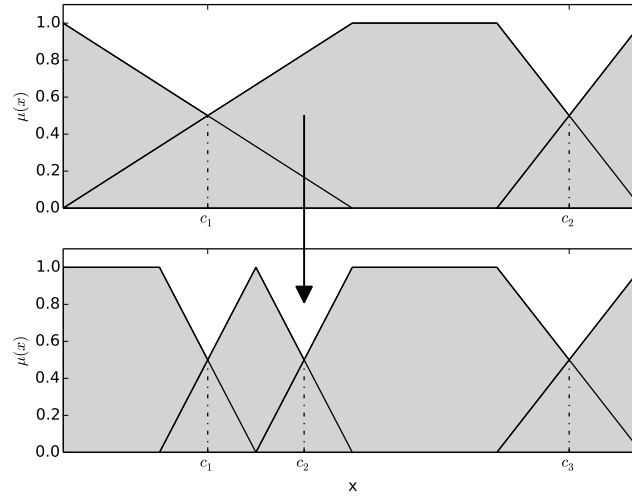


Figure 4: Top-down approach for the multi-granularity discretization. Only one label is divided into two new labels in order to obtain the next granularity.

In order to select the maximum number of split points for a variable, we used the well-known Bayesian Information Criterion (BIC). This criterion takes two issues into account: the error produced when applying the model to the data and its complexity. In our case, the error is obtained from the summation of the mean squared error ( $MSE$ ) of a least squares model fitted for each interval of the discretization. The complexity of the model is determined by the number of parameters, in this case the number of inner splits and the parameters fitted by each regression applied in each interval.

The pseudocode of the discretization method for a variable is shown in Fig. 5. First, the split points for granularity 1 are initialized using the domain limits (line 2). The BIC measure for this first granularity is calculated (line 3) using  $MSE$ , a function that gets a set of examples  $X$ , learns a linear regression model using least squares and, finally, calculates the mean squared error of the model. In this case, the number of parameters is two, corresponding to the coefficients of the linear model. After that, an iterative process is executed: at each step, the split points of a new granularity are defined adding a new split point to the previous granularity (lines 5-16).

---

```

1:  $g = 1$ 
2:  $C^g = \{\min(X), \max(X)\}$ 
3:  $BIC^g = |X| \cdot \log(MSE(X)) + 2 \cdot \log(|X|)$ 
4:  $it_{wi} = 0$ 
5: repeat
6:    $C = \{c_i \mid c_i = \operatorname{argmin}_c \text{LINEARERROR}(\{x \in X : C_i^g < x < C_{i+1}^g\}, c), \forall i = 0, \dots, g, \forall c \in [C_i^g, C_{i+1}^g]\}$ 
7:    $i_{min} = \operatorname{argmin}_i \text{LINEARERROR}(\{x \in X : C_i^g < x < C_{i+1}^g\}, c_i), \forall c_i \in C$ 
8:    $C^{g+1} = C^g \cup \{c_{i_{min}}\}$ 
9:    $g = g + 1$ 
10:   $BIC^g = |X| \cdot \log(\sum_{i=0}^g MSE(\{x \in X : C_i^g < x < C_{i+1}^g\})) + (|C^g| - 2) \cdot 2 \cdot \log(|X|)$ 
11:  if  $BIC^g < BIC^{min}$  then
12:     $it_{wi} = 0$ 
13:     $min = g$ 
14:  else
15:     $it_{wi} = it_{wi} + 1$ 
16: until  $it_{wi} > \sqrt{\frac{|X|}{30}} / min$ 
17: return  $C^1, \dots, C^{min}$ 

```

---

Figure 5: Pseudocode of the discretization method.

In order to obtain the split point for the new granularity, first, the best split point ( $c_i$ ) for each interval between the split points of the previous granularity ( $[C_i^g, C_{i+1}^g]$ ) is obtained (line 6). The best split point is defined as the point that obtains the global minimum of the function `LinearError` (Fig. 6) in an interval (Fig. 5, line 7). `LinearError` gets a set of examples  $X$  and a split point  $c$  and calculates the total squared error ( $SE$ ) of  $X$ , considering the corresponding linear regression models at each side of the split point. Only split points that obtain intervals with size of at least 30 are taken into account to assure that the obtained linear regressions are statistically valid.

The selected split point is added to the new granularity split points (lines 8-9), and the BIC measure is calculated (line 10). The number of parameters used for the BIC measure is 2 (coefficients of the linear regression for a single variable) for each interval. The number of intervals is calculated as  $|C^g| - 2$ , where 2 is subtracted to disregard the split points at the end of the domain of variable  $X$ .

---

```

1: function LINEARERROR( $X, c$ )
2:    $X_l = \{x \in X : x < c\}$ 
3:    $X_r = \{x \in X : x > c\}$ 
4:   return  $SE(X_l) \cdot \frac{|X_l|}{|X|} + SE(X_r) \cdot \frac{|X_r|}{|X|}$ 

```

---

Figure 6: Pseudocode of the function to be minimized by the discretization method.

Finally, when the number of consecutive iterations without improvement in the BIC value ( $it_{wi}$ ) is greater than  $\sqrt{\frac{|X|}{30}} / min$ , the algorithm stops (lines 11-16). This criterion ensures that at the beginning of the discretization process—the granularity is low—the BIC may worsen for more iterations, while with larger granularities, the algorithm becomes stricter in the stopping criterion. The number of data points is divided by 30 in order to get the maximum

number of intervals.

After obtaining the discretization of the variable for each granularity, the method proposed in [21] is applied for each  $C^g$  —set of split points for the granularity  $g$ — in order to get the multi-granularity fuzzy partitions. This method uses a fuzziness parameter: fuzziness 0 indicates crisp intervals, while fuzziness 1 indicates the selection of a fuzzy set with the smallest kernel —set of points with membership equal to 1.

### 3.3. Evolutionary Algorithm

The evolutionary algorithm learns a linguistic TSK model. The integration of the evolutionary algorithm with the preprocessing stage is as follows (Fig. 1):

- First, the instance selection process is executed over the training examples  $E_{tra}$  in order to obtain a subset of representative examples  $E_S$ .
- Then, the multi-granularity fuzzy discretization process obtains the fuzzy partitions for each input variable.
- Finally, the evolutionary algorithm searches for the best data base configuration using the obtained fuzzy partitions, generates the entire linguistic TSK rule base using  $E_S$  and evaluates the different rule bases using  $E_{tra}$ .

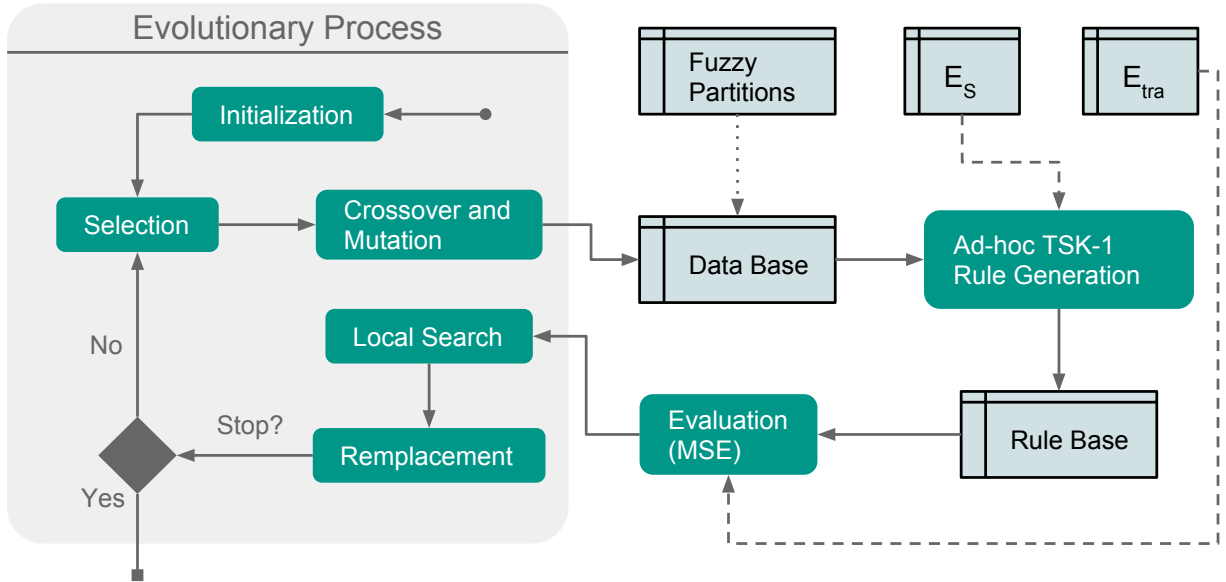


Figure 7: The Evolutionary learning process used in FRULER. Dashed lines indicate flow of datasets, dotted lines are for multigranularity information and solid lines represent process flow.

Figure 7 shows the evolutionary learning process and how it uses the fuzzy partitions and the training examples. In what follows, we describe in detail the different components of the evolutionary algorithm.

#### 3.3.1. Chromosome Codification

The chromosome codification represents the parameters needed to create the data base and the rule base. Each individual has to codify a single fuzzy partition for each input variable from the fuzzy partitions obtained in the multi-granularity fuzzy discretization (Sec. 3.2). Moreover, the individuals also use the 2-tuple representation of the labels [19]. This approach applies a displacement of a linguistic term within an interval that expresses the movement of a label between its two adjacent labels. In our case, a different displacement is going to be applied to each of the split points.

Thus, the chromosome is codified with a double coding scheme ( $C = C_1 + C_2$ ):



- $C_1$  represents the granularity of each input variable. It is codified with a vector of  $p$  integers:

$$C_1 = (g_1, g_2, \dots, g_p) \quad (6)$$

where  $g_i$  is the granularity of input variable  $i$ . When the granularity of a variable is equal to 1, then it is not used in the antecedent part. However, this variable can still be part of the consequent, since it may be relevant for calculating the output.

- $C_2$  represents the lateral displacements of the split points of the input variables fuzzy partitions. Thus, the length of  $C_2$  depends on the granularity of each input variable:  $|C_2| = \sum_{j=1}^p (|g_j| - 1), \forall g_j \in C_1$ :

$$C_2 = (\alpha_1^1, \dots, \alpha_1^{g_1-1}, \dots, \alpha_p^1, \dots, \alpha_p^{g_p-1}) \quad (7)$$

where  $\alpha_i^j$  is the lateral displacement of the  $j$  split point of variable  $i$ . Each lateral displacement can vary in the  $(0.5, 0.5)$  interval, which represents half of the distance between each split point (Fig. 8). An example of a lateral displacement can be seen in Figure 9. The fuzzy partitions are always strong —the sum of the degree of fulfillment for each point of the domain is always equal to 1— and, therefore, interpretability is maintained.

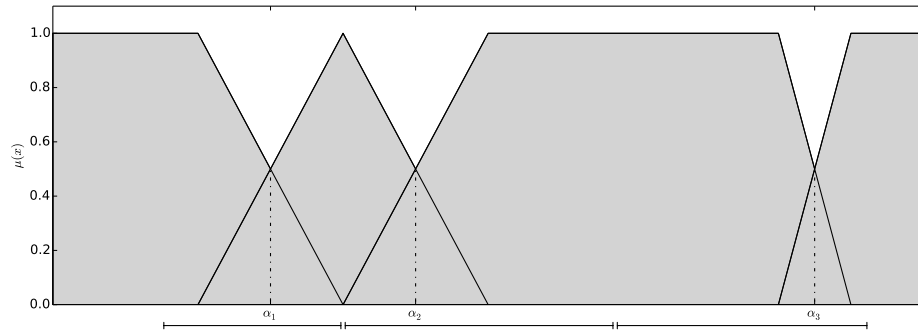


Figure 8: An example of lateral displacement intervals for limits equal to  $(0.5, 0.5)$ . The split points can move a maximum of half of the distance to the next split point.

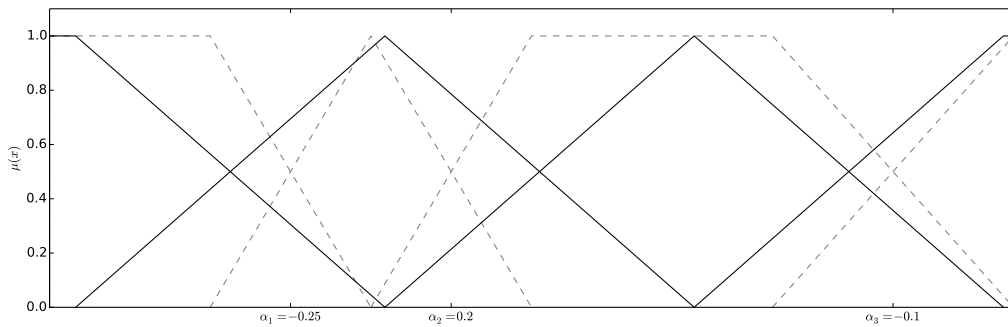


Figure 9: A lateral displacement example. The dashed lines indicate the original fuzzy partition, while the solid lines indicate the obtained partition after the displacement has been applied.

### 3.3.2. Initialization

The initial pool of individuals is generated by a combination of two initialization procedures. A half of the individuals are generated with the same random granularity for each variable, while the other half is created with a different random granularity for each variable. The lateral displacements are initialized to 0 in all cases.

After that, when the product of the granularities indicated in  $C_1$  (i.e., the maximum number of rules that can be obtained) is greater than the number of input variables times the highest maximum granularity of the variables, then a variable is randomly selected to be removed from the antecedent part—its granularity is set to 1—until the previous condition is satisfied. This is done in order to avoid too complex solutions in the initialization stage—during the evolutionary learning this upper bound to the number of rules does not apply.

### 3.3.3. TSK Rule Base Generation

An ad-hoc method is used to construct the rule base from the data base codified in the chromosome, i.e. the fuzzy partitions indicated in  $C_1$  after applying the displacement in  $C_2$ . The Wang & Mendel algorithm [39] is used to create the antecedent part of the rule base for each individual. The method is quick and simple, and obtains a representative rule base given the definition of the data base and a set of examples.

The consequent part of the rules is learned using the Elastic Net method [40] in order to obtain the coefficients of the degree 1 polynomial for each rule. Elastic Net linearly combines the  $\ell_1$  (Lasso regularization) and  $\ell_2$  (Ridge regularization) penalties of the Lasso and Ridge methods to overcome some of their limitations. This combination allows to obtain sparse models—forces variables with low or null correlation with the output to have coefficients equal to 0—while learning a smooth linear regression—the coefficients are shrunk towards 0.

Elastic Net obtains the coefficients of a linear regression minimizing the following equation:

$$\hat{\beta} = \arg \min_{\beta} \|Y - X \cdot \beta\|_2^2 + \lambda \cdot \alpha \cdot \|\beta\|_2^2 + \lambda \cdot (1 - \alpha) \cdot \|\beta\|_1 \quad (8)$$

where  $\beta$  is the coefficients vector  $(\beta_0, \beta_1, \dots, \beta_p)$ ,  $Y$  is the outputs vector  $(y^1, \dots, y^n)$ ,  $X$  is the inputs matrix with size  $n \times p$ —rows represent examples while columns are the input variables—,  $\lambda$  is the regularization parameter and  $\alpha$  represents the trade-off between  $\ell_1$  and  $\ell_2$  penalization.

In order to use Elastic Net for learning the consequents, the coefficients for each rule cannot be calculated separately due to the aggregation function used to obtain the output of the system (Eq. 3). Therefore, all the coefficients must be optimized at the same time, taking into account the degree of fulfillment of each rule (Eq. 2) for each input vector  $x^i$ . Thus, the matrix  $X$  is modified as follows:

- The normalized degree of fulfillment for each rule  $r_k$  and each example  $e^i$  is calculated as:

$$z_k^i = \frac{h_k(x^i)}{\sum_{u=1}^m h_u(x^i)} \quad (9)$$

where the denominator is the normalization term for each input vector  $x^i$ , i.e., the summation of the degree of fulfillment of all rules.

- Then, the matrix  $X$  is defined as:

$$X = \begin{pmatrix} z_1^1 & x_1^1 \cdot z_1^1 & \dots & x_p^1 \cdot z_1^1 & \dots & z_m^1 & x_1^1 \cdot z_m^1 & \dots & x_p^1 \cdot z_m^1 \\ \vdots & \vdots & & \vdots & & \vdots & \vdots & & \vdots \\ z_1^n & x_1^n \cdot z_1^n & \dots & x_p^n \cdot z_1^n & \dots & z_m^n & x_1^n \cdot z_m^n & \dots & x_p^n \cdot z_m^n \end{pmatrix} \quad (10)$$

where each row replicates the input vector  $x^i = (1, x_1^i, x_2^i, \dots, x_p^i)$ —a 1 was added to take into account the independent term—as many times as the number of rules ( $m$ ), weighting each rule  $r_k$  by  $z_k^i$ .

- Finally, the coefficient vector is the concatenation of the coefficients of all rules:

$$\beta = (\beta_0^1, \beta_1^1, \dots, \beta_p^1, \dots, \beta_0^m, \beta_1^m, \dots, \beta_p^m) \quad (11)$$

In order to solve the minimization problem of Elastic Net (Eq. 8), the Stochastic Gradient Descent (SGD) optimization technique was used [7, 37]. This gradient descent method is characterized by updating each coefficient separately using only one example at a time. This is particularly suited for sparse datasets, which is a common case when  $X$  is constructed using Eq. 10 —  $z_k^i$  is 0 when a rule does not cover an example.

The pseudocode of the method is shown in Figure 10. SGD needs three parameters to solve the Elastic Net approach: the regularization ( $\lambda$ ), the trade-off between Lasso and Ridge ( $\alpha$ ) and the initial learning rate ( $\eta^0$ ). On the one hand,  $\alpha$  usually takes a low value in order to behave like  $\ell_1$  but with the shrinkage of  $\ell_2$  in the features with coefficient not equal to 0. On the other hand,  $\lambda$  and  $\eta^0$  can be obtained using a grid search —testing a set of possible values in a predefined interval— using only a small subset of examples since the convergence properties are maintained [7].

The algorithm is composed of three different loops: i) lines 3-27, which represent an iteration over the whole dataset, ii) lines 6-17, which iterate over each example and iii) lines 11-17, which iterate over the coefficients. Note that, in this case, the number of coefficients is the number of columns in  $X$  ( $p \cdot m$ ), i.e., the number of input variables of the problem ( $p$ ) times the number of rules ( $m$ ). First, the examples are shuffled (line 5) each time the whole dataset is used. Then, for each example  $e^i$ ,  $t$  is incremented by 1 and the learning rate ( $\eta^t$ ) and the shrinkage portion for both  $\ell_1$  and  $\ell_2$  ( $s$  and  $u$  respectively) are updated (lines 6-10). After that, for each coefficient  $w_j$ , line 12 applies Ridge regularization [7], while lines 13-17 apply the Lasso approach [37]. The Lasso approach uses thresholds in order to decide if the variable is going to be selected (weight different from 0) and updates the threshold for each input variable for the next iterations ( $q_j$  in line 17). Finally, the coefficient of determination  $R^2$  is calculated (line 18) and compared with the best obtained so far. If it is better, then the estimated coefficients  $\hat{\beta}$  are updated and, if it is not, the number of iterations without improvement ( $it_{wi}$ ) is incremented by 1. When  $it_{wi}$  exceeds the threshold defined in line 27, the algorithm stops. This threshold is directly proportional to the number of examples, and decreases with the number of iterations.

Only those examples in  $E_s$  are used to obtain the rule base from the codified chromosome. In this manner, those examples that are not representative are not taken into account for the rule generation. Thus, the method avoids the generation of too specific rules, and reduces the time needed to create the rule base.

#### 3.3.4. Evaluation

The fitness function is based on the estimation of the error of the generated rule base:

$$fitness = MSE(E_{tra}) = \frac{1}{2 \cdot |E|} \sum_{i=1}^{|E|} (F(x^i) - y^i)^2, \quad (12)$$

where  $E_{tra}$  is the full training dataset and  $F(x^i)$  is the output obtained by the knowledge base for input  $x^i$ . Using all the examples for evaluation can be seen, in some way, as a validation process, as the rule base was constructed with a subset of them ( $E_s$ ).

#### 3.3.5. Selection and Replacement

The selection is performed by a binary tournament. On the other hand, the replacement method joins the previous and current populations, and selects the  $N$  best individuals as the new population.

#### 3.3.6. Crossover and Mutation

Two crossover operations are defined: one-point crossover for exchanging the  $C_1$  parts (it also exchanges the corresponding  $C_2$  genes) and, when the  $C_1$  parts are equal, the parent-centric BLX (PCBLX) [18] is used to crossover the  $C_2$  part. In order to prevent the crossover of too similar individuals, an incest prevention was implemented. When the euclidean distance of the lateral displacements is less than a particular threshold  $L$ , the individuals are not crossed.

The mutation (with probability  $p_{mut}$ ) applies two possible operations with equal probability to a randomly selected gene of the  $C_1$  part: i) decreasing the granularity by 1 or ii) increasing the granularity to a more specific granularity —all the granularities have the same chance. In order to calculate the new lateral displacements in the corresponding  $C_2$  part, the displacements of the previous granularity are taken into account. The displacement associated with a particular split point is calculated adding the displacements of the two nearest split points of the previous granularity (before mutation) weighted by the distance between the split points.

---

```

1: function SGD-ELASTICNET( $X, Y, \lambda, \alpha, \eta^0$ )
2:    $it = 0, t = 0, s = 1, u = 0, w^0 = 0_{1 \times p}, q = 0_{1 \times p}$ 
3:   repeat
4:     SHUFFLE_ROWS( $X$ ) ▷ SGD needs to reorder the rows of  $X$ 
5:     for  $i = 1, \dots, n$  do
6:        $t = t + 1$  ▷  $t$  counts how many updates of the weights have been applied
7:        $\eta^t = \eta^0 \cdot (\lambda \cdot t)^{-1}$  ▷ Updates the learning rate to be more conservative
8:        $\hat{y}^i = x^i \cdot w^t \cdot s$  ▷ Obtains the estimated output
9:        $s = s \cdot (1 - \alpha \cdot \eta^t \cdot \lambda)$  ▷ Updates how much of  $\ell_2$  was applied
10:       $u = u + (1 - \alpha) \cdot \eta^t \cdot \lambda$  ▷ Updates how much of  $\ell_1$  was applied
11:      for  $j = 1, \dots, p$  do
12:         $w_j^{t+\frac{1}{2}} = w_j^t - \eta^t \cdot (\hat{y}^i - y^i) \cdot x_j^i / s$  ▷ Applies the  $\ell_2$  regularization
13:        if  $s \cdot w_j^{t+\frac{1}{2}} > 0$  then ▷ Applies  $\ell_1$  regularization and thresholds in the following 5 lines
14:           $w_j^{t+1} = \max(0, w_j^{t+\frac{1}{2}} - (u + q_j) / s)$  ▷ Positive threshold
15:        else if  $s \cdot w_j^{t+\frac{1}{2}} < 0$  then
16:           $w_j^{t+1} = \min(0, w_j^{t+\frac{1}{2}} + (u - q_j) / s)$  ▷ Negative threshold
17:           $q_j = q_j + s \cdot (w_j^{t+1} - w_j^{t+\frac{1}{2}})$  ▷ Updates thresholds
18:         $R_t^2 = 1 - \frac{1}{n} \sum_{i=0}^n (x^i \cdot w^{t+1} \cdot s - y^i)^2$  ▷ Calculates the coefficient of determination
19:        if  $R_t^2 > R_{best}^2$  then ▷ Updates the best values so far and the iterations without improvement
20:           $\hat{\beta} = w^{t+1} \cdot s$ 
21:           $R_{best}^2 = R_t^2$ 
22:           $it_{wi} = 0$ 
23:        else
24:           $it_{wi} = it_{wi} + 1$ 
25:         $it = it + 1$  ▷  $it$  counts how many times the full dataset was used
26:      until  $it_{wi} > \text{sqrt}(|X|/it)$ 
27:      return  $\hat{\beta}$ 

```

---

Figure 10: Pseudocode of SGD for Elastic-Net.

### 3.3.7. Local Search

After the replacement, all the new individuals (their  $C_1$  part of the chromosome was not generated before) go to a local search process. This stage generates  $n_{ls}$  new  $C_1$  parts with equal or less granularity—with equal probability—for each variable. Then, the  $C_2$  part is generated randomly with a uniform distribution in the  $(-0.5, 0.5)$  interval. The new chromosomes are decoded and evaluated and, if there is a solution that obtains better fitness, then it replaces the original individual.

### 3.3.8. Restart and Stopping Criteria

The restart mechanism uses the incest prevention threshold  $L$  as a trigger. First,  $L$  is initialized as the maximum length of the  $C_2$  part, i.e. the product of the number of input variables times the largest maximum granularity of the variables, divided by 4. This implies that the incest prevention allows crossovers between individuals that have a distance higher than a quarter of the maximum euclidean distance. Then, at each iteration,  $L$  is decreased in different ways in order to accelerate convergence:

- $L$  is decreased by 0.4 in all the iterations, in order to increase convergence.
- If there are no new individuals in the population, then  $L$  is decreased by 0.2.

- If the best individual does not change,  $L$  is also decreased by 0.2.

Finally, when  $L$  reaches 0, the population is restarted, and  $L$  is reinitialized. Only the best individual so far is kept, and the local search process is executed with the best individual in order to generate new individuals until the population is complete. When the restart criterion is fulfilled twice, the algorithm stops, i.e., one single restart is executed. Moreover, if the number of evaluations reaches a threshold, then the algorithm is also stopped. When the evolutionary algorithm stops, the best rule base consequents are optimized applying the SGD algorithm (Sec. 3.3.3) using all the training examples.

#### 4. Results

In order to analyze the performance of FRULER, we have used 28 real-world regression problems from the KEEL project repository [3]. Table 1 shows the characteristics of the datasets, with the number of instances ranging from 337 to 40,768 examples, and the number of input variables from 2 to 40. The most complex problems —large scale— due to both the number of examples and variables are the ones in the last 8 rows (Table 1).

Problem	Abbr.	#Variables	#Cases
Electrical Length	ELE1	2	495
Plastic Strength	PLA	2	1,650
Quake	QUA	3	2,178
Electrical Maintenance	ELE2	4	1,056
Friedman	FRIE	5	1,200
Auto MPG6	MPG6	5	398
Delta Ailerons	DELAILE	5	7,129
Daily Electricity Energy	DEE	6	365
Delta Elevators	DELELV	6	9,517
Analcat	ANA	7	4,052
Auto MPG8	MPG8	7	398
Abalone	ABA	8	4,177
Concrete Compressive Strength	CON	8	1,030
Stock prices	STP	9	950
Weather Ankara	WAN	9	1,609
Weather Izmir	WIZ	9	1,461
Forest Fires	FOR	12	517
Mortgage	MOR	15	1,049
Treasury	TRE	15	1,049
Baseball	BAS	16	337
California Housing	CAL	8	20,640
MV Artificial Domain	MV	10	40,768
House-16H	HOU	16	22,784
Elevators	ELV	18	16,559
Computer Activity	CA	21	8,192
Pole Telecommunications	POLE	26	14,998
Pumadyn	PUM	32	8,192
Ailerons	AIL	40	13,750

Table 1: The 28 datasets of the experimental study stating their number of input variables (#Variables) and examples (#Cases).

In the following subsections we show the results obtained by the different parts of the algorithm. Moreover, the results obtained by FRULER are compared with other state of the art approaches.

#### 4.1. Experimental Setup

FRULER was designed to keep the number of parameters as low as possible. For the instance selection technique, no parameters are needed. In the multi-granularity fuzzy discretization stage, the fuzziness parameter for the generation of the fuzzy intervals was set to 1, i.e., the highest fuzziness value. For the evolutionary algorithm, the values of the parameters were: population size = 61, maximum number of evaluations = 100,000,  $p_{cross} = 1.0$ ,  $p_{mut} = 0.2$ , and  $n_{ls} = 5$ . For the generation of the TSK fuzzy rule bases, the weight of the tradeoff between  $\ell_1$  and  $\ell_2$  regularizations of Elastic Net was  $\alpha = 0.95$ , and the regularization parameter  $\lambda$  was obtained from a grid search in the interval  $[1, 1E - 10]$ .  $\eta^0$  was obtained halving the initial value (0.1) until the result worsens.

A 5-fold cross validation was used in all the experiments. Moreover, 6 trials (with different seeds for the random number generation) of FRULER were executed for each 5-fold cross validation. Thus, a total of 30 runs were obtained for each dataset. The results shown in the next section are the mean values over all the runs. The run times were measured using a single thread in an Intel Xeon Processor E5-2650L (20M Cache, 1.80 GHz, 8.00 GT/s Intel QPI).

#### 4.2. Performance of the Instance Selection Process

We considered two different measures to evaluate the instance selection process:

- Reduction: is the percentage of reduction in the number of examples, defined as:

$$Reduction = \left(1 - \frac{|E_s|}{|E_{tra}|}\right) \cdot 100 \quad (13)$$

where  $|E_s|$  is the number of examples in the subset of selected examples and  $|E_{tra}|$  is the original number of examples in the training set.

- Increase in error: is the increment in the error after applying the instance selection process, defined as:

$$Increase = \frac{\epsilon_{E_s}}{\epsilon_{E_{tra}}} \quad (14)$$

where  $\epsilon_E$  is the mean squared error of a leave-one-out 1NN regression.

Table 2 shows the average values of instances reduction and error increase for each dataset. The percentage of reduction achieved is over 80% in most of the datasets. In four of the datasets (QUA, FRIE, DEE, STP) the reduction is in the range 70-80%, and only one dataset (WIZ) has a reduction under 70% (64.2%). The reduction rate does not depend neither on the size of the dataset, nor on the number of variables, but on the complexity of the data. On the other hand, the increase in 1NN error is very low, as it is greater than 2 for only eight datasets (ELE2, ANA, STP, MOR, TRE, MV, CA, POLE). The run time of the instance selection process is generally low, and only the large scale problems take more than 15 minutes.

#### 4.3. Performance of the Multi-Granularity Fuzzy Discretization Process

We evaluated the discretization with three different measures:

- Average maximum granularity (over all the variables) for each dataset. This measure summarizes the complexity of the fuzzy partitions generated by the discretization.
- Maximum granularity among the variables for each dataset. This represents the upper bound of the fuzzy partitions obtained for each dataset. It is expected that the smaller this value, the simpler the models obtained by FRULER.
- The number of variables that have not been discretized at all, i.e., their maximum granularity is equal to 1.

Table 3 summarizes the results for each dataset. The average maximum granularity is below 9 in all the cases except for DELAIL dataset. Moreover, the maximum granularity is always below 20 and only in 11 cases (DELAIL, DELELV, CON, WAN, CAL, MV, HOU, ELE, CA, POLE, AIL) it is above granularity 10. Even in the datasets with high granularities, the maximum number of fuzzy sets does not generate a huge search space for the evolutionary algorithm. Finally, the number of variables without discretization is 0 in most of the cases. In terms of run time, the discretization module has almost no cost, as the most expensive discretization process is less than 3 seconds.

Datasets	Reduction (%)	Increase in error	Time (m:s)
ELE1	83.4	0.85	00:54
PLA	91.8	0.76	01:54
QUA	70.9	1.07	03:50
ELE2	84.9	4.08	02:09
FRIE	79.5	1.53	01:33
MPG6	81.6	1.24	00:45
DELAAIL	96.9	1.04	10:59
DEE	77.7	1.14	00:42
DELELV	94.0	0.97	15:44
ANA	98.8	16.61	05:09
MPG8	81.4	0.92	00:44
ABA	91.7	0.96	06:13
CON	88.0	1.33	02:04
STP	73.6	2.35	02:07
WAN	85.4	1.58	02:03
WIZ	64.2	1.36	02:37
FOR	93.3	0.54	00:58
MOR	83.4	4.28	02:12
TRE	81.6	5.35	02:20
BAS	83.5	1.61	00:38
CAL	91.6	1.27	39:44
MV	98.7	3.87	40:33
HOU	95.4	1.12	46:08
ELE	96.0	1.33	30:49
CA	98.9	7.40	11:57
POLE	98.7	18.36	27:15
PUM	80.3	1.01	14:48
AIL	95.4	1.12	24:31

Table 2: Average (5-fold cross validation) results of reduction (percentage of reduction in the number of examples), error increase (increment in the error after applying the instance selection process) and runtime obtained by the instance selection method for each dataset.

#### 4.4. Statistical Analysis

In this section we compare FRULER with three of the most accurate genetic fuzzy systems for regression in the literature:

- $FS_{MOGFS}^e + TUN^e$  [2]: a multi-objective evolutionary algorithm that learns Mamdani fuzzy rule bases. This algorithm learns the granularities from uniform multi-granularity fuzzy partitions (up to granularity 7) and the lateral displacement of the labels. It includes a post-processing algorithm for tuning the parameters of the membership functions and for rule selection.
- L-METSK-HD<sup>e</sup> [15]: a multi-objective evolutionary algorithm that learns linguistic TSK-0 fuzzy rule bases. The algorithm learns the granularities from uniform multi-granularity fuzzy partitions (up to granularity 7).
- A-METSK-HD<sup>e</sup> [15]: a multi-objective evolutionary algorithm that learns approximative TSK-1 fuzzy rule bases. The algorithm starts with the solution obtained on the first stage and applies a tuning of the membership functions, rule selection and a Kalman-based calculation of the consequents of the rules.

Table 4 shows the average results of FRULER and the three algorithms selected for comparison. Two different results are shown for each algorithm and dataset: the number of rules of the obtained rule base, and the test error measured using equation 12 over the test data. These indicators allow to compare both the simplicity and the accuracy

Problem	Average	Max	#Not used	Time (s:ms)
ELE1	2.3	2.4	0.0	00:11
PLA	3.5	4.4	0.0	00:17
QUA	3.9	7.8	0.0	00:26
ELE2	5.1	7.6	0.0	00:21
FRIE	2.0	2.0	0.0	00:18
MPG6	3.6	5.8	0.0	00:13
DELAAIL	9.8	14.0	0.0	00:65
DEE	2.2	3.0	0.0	00:11
DELELV	7.8	15.4	0.8	00:63
ANA	2.0	6.6	5.0	00:13
MPG8	2.9	5.6	1.0	00:09
ABA	4.0	7.8	1.0	00:35
CON	6.0	14.0	1.0	00:29
STP	4.3	9.4	0.0	00:24
WAN	5.0	14.8	0.0	00:28
WIZ	4.3	9.0	0.0	00:25
FOR	2.4	6.0	4.0	00:15
MOR	3.9	8.0	0.0	00:27
TRE	3.7	6.2	0.0	00:35
BAS	2.6	5.2	4.0	00:13
CAL	5.1	13.8	0.0	01:40
MV	3.4	18.6	3.0	02:91
HOU	3.7	12.2	5.0	02:20
ELE	8.0	17.2	2.0	01:57
CA	4.1	14.4	8.0	00:73
POLE	4.5	16.0	5.0	01:05
PUM	2.0	3.2	0.0	01:41
AIL	6.7	19.0	6.2	02:01

Table 3: Results of the multi-granularity fuzzy discretization process for each dataset —5-fold cross validation. The table shows the average granularity of all the input variables (Average), the maximum granularity (Max), the number of input variables with granularity 1 (#Not Used), and the runtime (Time).

of the learned models. The values with the best accuracy —lowest error— and best number of rules in table 4 are marked in bold.

It can be seen that the number of rules of FRULER is the lowest in the majority of the datasets. It should be noted that the number of rules in the large scale problems (the last 8 problems) is also low despite the high number of examples. Only in 5 problems the  $FS_{MOGFS}^e + TUN^e$  Mamdani proposal produces the lowest number of rules. In the case of accuracy, in 15 of the 28 problems FRULER achieves the best results. In the other 13 datasets, the best results are for  $FS_{MOGFS}^e + TUN^e$  (best in 4 problems) and A-METSK-HD<sup>e</sup> (best in 9 problems). From the results, we did not find any influence in the performance of FRULER by neither the training dataset size nor the dimensionality of the problem.

In order to analyze the statistical significance of these results, we used the STAC platform [28] to apply the statistical tests. A Friedman test was used for both the number of rules and the test error in order to get a ranking of the algorithms and check whether the differences between them were statistically significant.

Table 5 shows the ranking for the test error, with the p-value of the test. Our proposal —generates linguistic TSK-1 rules— gets the top ranking, i.e., it has the best results in accuracy among all the algorithms. Then, the next algorithm in the ranking is the approximative approach, due to its fine tuning of the rules, followed by the linguistic approaches. In order to compare whether the difference between FRULER and the second ranked algorithm (A-METSK-HD<sup>e</sup>[15]) was significant, we performed a Wilcoxon test (Table 6). The p-value indicates that the difference is statistically



algorithms	FRULER		FS <sub>MOGFS</sub> <sup>e</sup> +TUN <sup>e</sup>		L-METSK-HD <sup>e</sup>		A-METSK-HD <sup>e</sup>	
	#Rules	Test Error	#Rules	Test Error	#Rules	Test Error	#Rules	Test Error
ELE1	<b>4.1</b>	2.012	8.1	<b>1.954</b>	15	1.925	11.4	2.022
PLA	<b>1.4</b>	1.219	18.6	1.194	23	1.218	19.2	<b>1.136</b>
QUA	7.8	0.0181	<b>3.2</b>	<b>0.0178</b>	35.9	0.019	18.3	0.0181
ELE2	<b>4.3</b>	6,729	8	10,548	59	20,095	36.9	<b>3,192</b>
FRIE	<b>8.0</b>	<b>0.731</b>	22	3.138	95.1	3.084	66	1.888
MPG6	<b>13.7</b>	<b>3.727</b>	20	4.562	99.6	4.469	53.6	4.478
DELAILE	<b>2.5</b>	1.458	6.2	1.528	98.3	1.621	36.8	<b>1.402</b>
DEE	<b>7.9</b>	<b>0.080</b>	18.3	0.093	96.4	0.095	50.6	0.103
DELELV	<b>5.8</b>	1.045	7.9	1.086	91	1.119	39.1	1.031
ANA	<b>3.9</b>	0.008	10	<b>0.003</b>	48.9	0.006	33.3	0.004
MPG8	<b>12.7</b>	<b>4.084</b>	23	4.747	98.7	5.61	64.2	5.391
ABA	<b>4.5</b>	2.393	8	2.509	42.4	2.581	23.1	<b>2.392</b>
CON	<b>8.9</b>	<b>20.598</b>	15.4	32.977	96.5	38.394	53.7	23.885
STP	42.4	<b>0.353</b>	<b>23</b>	0.912	100	0.78	66.4	0.387
WAN	<b>5.6</b>	<b>0.888</b>	8	1.635	91.1	1.773	48	1.189
WIZ	<b>8.9</b>	<b>0.663</b>	10	1.011	55.4	1.296	29.1	0.944
FOR	<b>5.6</b>	<b>2,214</b>	10	2,628	93.7	4,633	40.6	5,587
MOR	7.9	<b>0.007</b>	7	0.019	40.9	0.028	27.2	0.013
TRE	<b>4.5</b>	<b>0.027</b>	9	0.044	42.8	0.052	28.1	0.038
BAS	<b>6.2</b>	305,777	17	<b>261,322</b>	95.7	320,133	59.8	368,820
CAL	15.4	2.110	<b>8.4</b>	2.95	99.8	2.638	55.8	<b>1.71</b>
MV	<b>6.0</b>	0.083	14	0.158	76.4	0.244	56.5	<b>0.061</b>
HOU	12.1	<b>8.005</b>	<b>11.7</b>	9.4	68.9	10.368	30.5	8.64
ELE	<b>5.4</b>	<b>2.934</b>	8	9	76.4	8.9	34.9	7.02
CA	<b>7.1</b>	<b>4.634</b>	14	5.216	71.3	5.88	32.9	4.949
POLE	40.8	110.898	<b>13.1</b>	102.816	100	150.673	46.3	<b>61.018</b>
PUM	<b>7.8</b>	0.367	17.6	0.292	87.5	0.594	63.3	<b>0.287</b>
AIL	<b>8.5</b>	<b>1.404</b>	15	2	99.1	1.822	48.4	1.51

Table 4: Average number of rules (#Rules) and test MSE (Test Error) for the compared algorithms. The test errors in this table should be multiplied by  $10^5$ ,  $10^{-8}$ ,  $10^{-6}$ ,  $10^9$ ,  $10^8$ ,  $10^{-6}$ ,  $10^{-4}$ ,  $10^{-8}$  in the case of ELE1, DELAILE, DELELV, CAL, HOU, ELV, PUM, AIL respectively.

Algorithm	Ranking
FRULER	1.714
A-METSK-HD <sup>e</sup>	2.036
FS <sub>MOGFS</sub> <sup>e</sup> +TUN <sup>e</sup>	2.786
L-METSK-HD <sup>e</sup>	3.464
p-value	$< 1E - 5$

Table 5: Friedman test ranking results and test p-value for the test error in table 4.

Comparison	p-value
FRULER vs A-METSK-HD <sup>e</sup>	0.079

Table 6: Wilcoxon comparison for the two most accurate algorithms in table 5.

significant when using a significance level of 0.1. Thus, even with linguistic rules, FRULER obtains a great accuracy compared to approximative approaches, while getting simpler models.

To analyze the complexity of the models obtained for each algorithm, the same Friedman test was performed to the number of rules in table 4 (Table 7). Once again, FRULER has the lowest ranking. The next algorithm in the

Algorithm	Ranking
FRULER	1.214
FS <sub>MOGFS</sub> <sup>e</sup> +TUN <sup>e</sup>	1.786
A-METSK-HD <sup>e</sup>	3
L-METSK-HD <sup>e</sup>	4
p-value	$< 1E - 5$

Table 7: Friedman test ranking and test p-value results for the number of rules in table 4.

Comparison	p-value
FRULER vs A-METSK-HD <sup>e</sup>	$< 1E - 4$

Table 8: Wilcoxon comparison of the number of rules for the two most accurate approaches in table 7.

ranking is FS<sub>MOGFS</sub><sup>e</sup>+TUN<sup>e</sup> Mamdani approach, followed by the METSK-HD<sup>e</sup> approaches with a big difference in the ranking. We applied a Wilcoxon test (Table 8) in order to assess whether the difference in complexity among the most accurate proposals (Table 5) was significant. The difference is statistically significant (p-value equal to  $1E - 4$ ) in the number of rules. This shows that FRULER obtains accurate solutions with simpler models.

Datasets	ELE1	PLA	QUA	ELE2	FRIE	MPG6	DELAILE	DEE	DELELV	ANA
Time (h:m:s)	0:00:51	0:01:41	0:09:48	0:03:05	0:05:46	0:02:12	0:09:58	0:02:26	0:25:01	0:05:05
Evaluations	8,885	7,345	13,020	16,798	17,283	21,556	19,236	24,131	24,386	27,107
Datasets	MPG8	ABA	CON	STP	WAN	WIZ	FOR	MOR	TRE	BAS
Time (h:m:s)	0:03:29	0:17:45	0:05:55	0:27:41	0:10:27	0:26:03	0:02:28	0:27:50	0:23:30	0:04:41
Evaluations	29,355	31,537	32,318	38,468	35,812	36,168	45,367	60,101	57,569	59,362
Datasets	CAL	MV	HOU	ELE	CA	POLE	PUM	AIL		
Time (h:m:s)	1:57:03	1:17:02	4:15:17	3:01:30	0:38:12	1:53:15	31:14:27	12:50:38		
Evaluations	33,951	35,001	61,709	68,055	78,036	99,827	96,543	100,000		

Table 9: Average runtime and number of evaluations per run of FRULER.

Table 9 shows the average run time of FRULER in each dataset. We also display the number of evaluations until the stopping condition was met. Although each of the stages of FRULER increases the computational complexity, they contribute to focus the search on the simplest models. Our method obtains solutions in the range between 1-23 minutes for datasets 1-20 (the most simple ones) and solutions in the range from 1-30 hours for datasets 21-28 (the most complex ones). Moreover, the number of evaluations is below the limit (100,000), except for the largest problem (AIL). The run time of FRULER is in the same order of magnitude as A-METSK-HD, being only worse in six datasets (QUA, WIZ, MOR, TRE, PUM and AIL) <sup>1</sup>.

In order to demonstrate the simplicity of the models generated by FRULER, Figure 11 shows an example of one of the rule bases generated for the WAN dataset. There are two columns for each rule: the fuzzy sets in the antecedent and the weight of the variables in the consequent. For the sake of simplicity and understandability, the consequents are represented with their absolute value and have been scaled to a maximum weight of 1. The antecedent only uses two variables with granularity 3 and 2 respectively, thus 6 rules are needed to cover all the combinations. On the other hand, the consequent column shows the importance of each input variable for each rule, providing a qualitative understanding of the model. In this case, the first three variables (X1, X2 and X3) have the greatest importance in the

<sup>1</sup>A quantitative comparison with the computational times of [15] has not been performed, since the processors are different

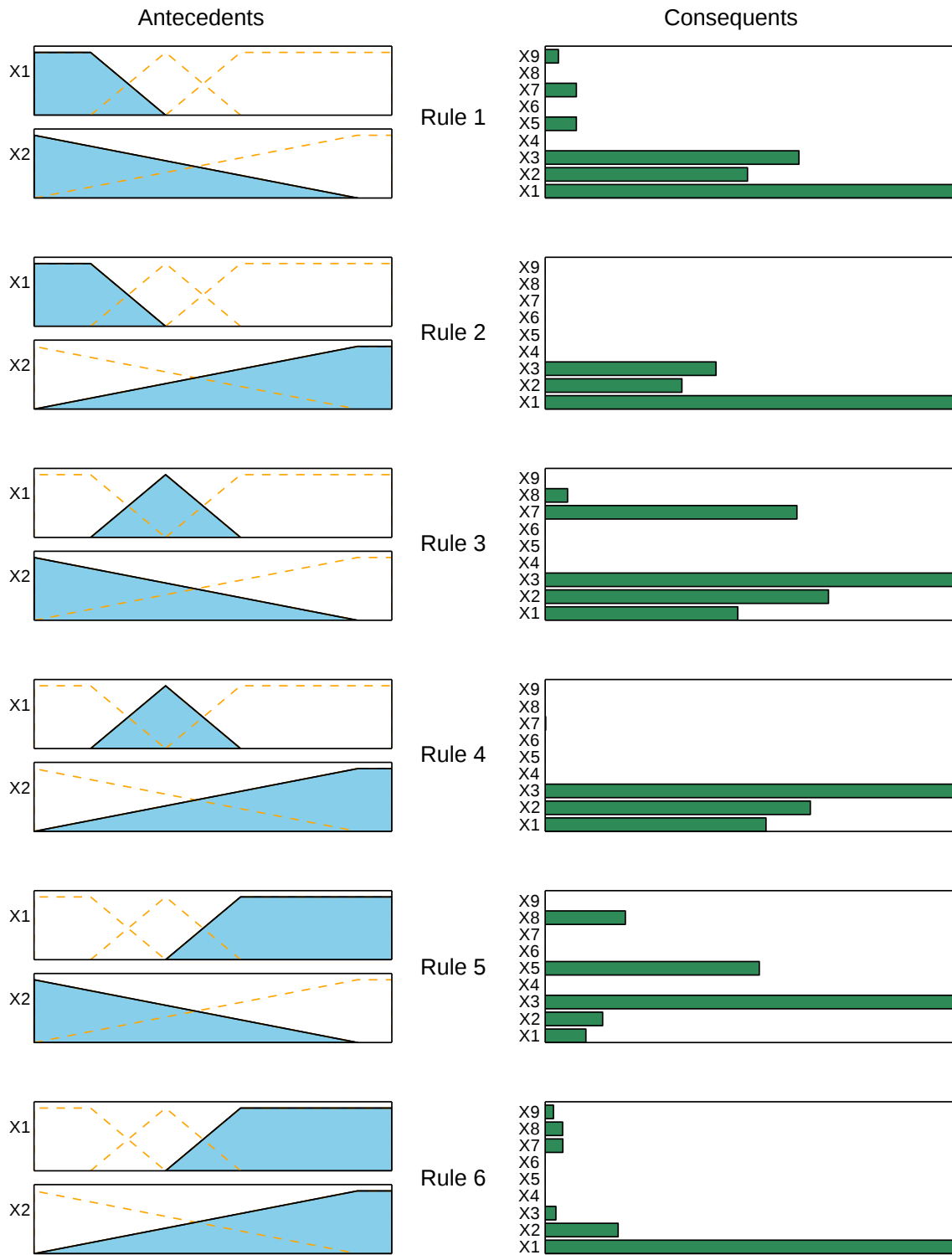


Figure 11: An example of TSK fuzzy rule base for the WAN dataset. The system uses only 2 variables for the antecedent part and has 6 different rules. For the sake of simplicity and understandability, the consequents are represented with their absolute value and have been scaled to have the maximum weight equal to 1. The test error obtained by this example is 0.885.

consequent. Note that, even though this is one of the simplest models obtained by FRULER, the test error is very low (0.885).

## 5. Conclusions

In this paper, a novel genetic fuzzy system called FRULER was presented. FRULER learns simple and linguistic TSK-1 knowledge bases for regression problems following a new approach than involves two general-purpose preprocessing stages: a new instance selection for regression and a novel non-uniform multi-granularity fuzzy discretization. Furthermore, FRULER's evolutionary learning algorithm incorporates an automatic generation of the TSK fuzzy rule bases from fuzzy partitions, and uses Elastic Net in order to obtain consequents with low overfitting.

FRULER was compared with three state of the art algorithms that learn different types of fuzzy rules: linguistic Mamdani, linguistic TSK-0 and approximative TSK-1. The results were analyzed using statistical tests of significance, which show that FRULER obtains high accuracy, but with a lower number of rules and with a linguistic data base. This is of particular interest in problems where both high accuracy and interpretability are demanded, in order to provide simple and understandable regression models to the users.

## Acknowledgments

This work was supported by the Spanish Ministry of Economy and Competitiveness under projects TIN2011-22935, TIN2011-29827-C02-02 and TIN2014-56633-C3-1-R, and the Galician Ministry of Education under the projects EM2014/012 and CN2012/151. I. Rodríguez-Fdez is supported by the Spanish Ministry of Education, under the FPU national plan (AP2010-0627).

## References

- [1] Alcalá, R., Alcalá-Fdez, J., Herrera, F., Otero, J., 2007. Genetic learning of accurate and compact fuzzy rule based systems based on the 2-tuples linguistic representation. *International Journal of Approximate Reasoning* 44 (1), 45–64.
- [2] Alcalá, R., Gacto, M. J., Herrera, F., 2011. A fast and scalable multiobjective genetic fuzzy system for linguistic fuzzy modeling in high-dimensional regression problems. *IEEE Transactions on Fuzzy Systems* 19 (4), 666–681.
- [3] Alcalá-Fdez, J., Sánchez, L., García, S., del Jesus, M. J., Ventura, S., Garrell, J., Otero, J., Romero, C., Bacardit, J., Rivas, V. M., et al., 2009. Keel: a software tool to assess evolutionary algorithms for data mining problems. *Soft Computing* 13 (3), 307–318.
- [4] Alonso, J. M., Castiello, C., Mencar, C., 2015. Interpretability of fuzzy systems: Current research trends and prospects. In: *Springer Handbook of Computational Intelligence*. Springer, pp. 219–237.
- [5] Alonso, J. M., Magdalena, L., 2011. Special issue on interpretable fuzzy systems. *Information Sciences* 181 (20), 4331–4339.
- [6] Antonelli, M., Ducange, P., Marcelloni, F., 2013. An efficient multi-objective evolutionary fuzzy system for regression problems. *International Journal of Approximate Reasoning* 54 (9), 1434–1451.
- [7] Bottou, L., 2010. Large-scale machine learning with stochastic gradient descent. In: *Proceedings of the 19th International Conference on Computational Statistics*. Springer, pp. 177–186.
- [8] Casillas, J., Cordon, O., Triguero, F. H., Magdalena, L., 2013. Accuracy improvements in linguistic fuzzy modeling. Vol. 129. Springer.
- [9] Casillas, J., Cordon, O., Triguero, F. H., Magdalena, L., 2013. Interpretability issues in fuzzy modeling. Vol. 128. Springer.
- [10] Cordon, O., Herrera, F., Hoffmann, F., Magdalena, L., Cordon, O., Herrera, F., Hoffmann, F., 2001. Genetic fuzzy systems. World Scientific Publishing Company Singapore.
- [11] Dougherty, J., Kohavi, R., Sahami, M., et al., 1995. Supervised and unsupervised discretization of continuous features. In: *Machine learning: proceedings of the twelfth international conference*. Vol. 12. pp. 194–202.
- [12] Fazzolari, M., Alcalá, R., Nojima, Y., Ishibuchi, H., Herrera, F., 2013. A review of the application of multiobjective evolutionary fuzzy systems: Current status and further directions. *IEEE Transactions on Fuzzy Systems* 21 (1), 45–65.
- [13] Fazzolari, M., Alcalá, R., Herrera, F., 2014. A multi-objective evolutionary method for learning granularities based on fuzzy discretization to improve the accuracy-complexity trade-off of fuzzy rule-based classification systems: D-mofarc algorithm. *Applied Soft Computing* 24, 470–481.
- [14] Fazzolari, M., Giglio, B., Alcalá, R., Marcelloni, F., Herrera, F., 2013. A study on the application of instance selection techniques in genetic fuzzy rule-based classification systems: Accuracy-complexity trade-off. *Knowledge-Based Systems* 54, 32–41.
- [15] Gacto, M. J., Galende, M., Alcalá, R., Herrera, F., 2014. Metsk-hd e: A multiobjective evolutionary algorithm to learn accurate tsk-fuzzy systems in high-dimensional and large-scale regression problems. *Information Sciences* 276, 63–79.
- [16] García, S., Derrac, J., Cano, J. R., Herrera, F., 2012. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34 (3), 417–435.
- [17] García, S., Luengo, J., Sáez, J. A., López, V., Herrera, F., 2013. A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. *IEEE Transactions on Knowledge and Data Engineering* 25 (4), 734–750.

- [18] Herrera, F., Lozano, M., Sánchez, A. M., 2003. A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study. *International Journal of Intelligent Systems* 18 (3), 309–338.
- [19] Herrera, F., Martínez, L., 2000. A 2-tuple fuzzy linguistic representation model for computing with words. *IEEE Transactions on Fuzzy Systems* 8 (6), 746–752.
- [20] Ishibuchi, H., Nojima, Y., 2007. Analysis of interpretability-accuracy tradeoff of fuzzy systems by multiobjective fuzzy genetics-based machine learning. *International Journal of Approximate Reasoning* 44 (1), 4–31.
- [21] Ishibuchi, H., Yamamoto, T., 2002. Performance evaluation of fuzzy partitions with different fuzzification grades. In: *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. Vol. 2. IEEE, pp. 1198–1203.
- [22] Mamdani, E. H., 1974. Application of fuzzy algorithms for control of simple dynamic plant. In: *Proceedings of the Institution of Electrical Engineers*. Vol. 121. IET, pp. 1585–1588.
- [23] Mamdani, E. H., Assilian, S., 1975. An experiment in linguistic synthesis with a fuzzy logic controller. *International journal of man-machine studies* 7 (1), 1–13.
- [24] Marchiori, E., 2010. Class conditional nearest neighbor for large margin instance selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32 (2), 364–370.
- [25] Márquez, A. A., Márquez, F. A., Roldán, A. M., Peregrín, A., 2013. An efficient adaptive fuzzy inference system for complex and high dimensional regression problems in linguistic fuzzy modelling. *Knowledge-Based Systems* 54, 42–52.
- [26] Mucientes, M., Rodríguez-Fdez, I., Bugarín, A., 2009. Evolutionary learning of quantified fuzzy rules for hierarchical grouping of laser sensor data in intelligent control. In: *Proceedings of the IFSA-EUSFLAT 2009 conference*. Lisbon (Portugal), pp. 1559–1564.
- [27] Mucientes, M., Vidal, J. C., Bugarín, A., Lama, M., 2009. Processing time estimations by variable structure tsf rules learned through genetic programming. *Soft Computing* 13 (5), 497–509.
- [28] Rodríguez-Fdez, I., Canosa, A., Mucientes, M., Bugarín, A., 2015. STAC: a web platform for the comparison of algorithms using statistical tests. In: *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. pp. 1–8.
- [29] Rodríguez-Fdez, I., Mucientes, M., Bugarín, A., 2011. Iterative rule learning of quantified fuzzy rules for control in mobile robotics. In: *Proceedings of IEEE Symposium Series on Computational Intelligence (SSCI)*. Paris (France), pp. 111–118.
- [30] Rodríguez-Fdez, I., Mucientes, M., Bugarín, A., 2012. Photons detection in positron emission tomography through iterative rule learning of tsf rules. In: *Actas del VIII Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB)*. Albacete (Spain), pp. 251–258.
- [31] Rodríguez-Fdez, I., Mucientes, M., Bugarín, A., 2013. An instance selection algorithm for regression and its application in variance reduction. In: *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. pp. 1–8.
- [32] Rodríguez-Fdez, I., Mucientes, M., Bugarín, A., 2015. Learning fuzzy controllers in mobile robotics with embedded preprocessing. *Applied Soft Computing* 26, 123–142.
- [33] Sánchez, L., Otero, J., Couso, I., 2009. Obtaining linguistic fuzzy rule-based regression models from imprecise data with multiobjective genetic algorithms. *Soft Computing* 13 (5), 467–479.
- [34] Scott, D. W., 2009. *Multivariate density estimation: theory, practice, and visualization*. Vol. 383. John Wiley & Sons.
- [35] Sugeno, M., Kang, G., 1988. Structure identification of fuzzy model. *Fuzzy sets and systems* 28 (1), 15–33.
- [36] Takagi, T., Sugeno, M., 1985. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics* (1), 116–132.
- [37] Tsuruoka, Y., Tsujii, J., Ananiadou, S., 2009. Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In: *Proceedings of the Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*. Association for Computational Linguistics, pp. 477–485.
- [38] Vidal, J. C., Mucientes, M., Bugarín, A., Lama, M., 2011. Machine scheduling in custom furniture industry through neuro-evolutionary hybridization. *Applied Soft Computing* 11 (2), 1600–1613.
- [39] Wang, L.-X., Mendel, J. M., 1992. Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man and Cybernetics* 22 (6), 1414–1427.
- [40] Zou, H., Hastie, T., 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society* 67 (2), 301–320.