frbs: Fuzzy Rule-Based Systems for Classification and Regression in R

Lala Septem Riza
University of Granada

Christoph Bergmeir
University of Granada

Francisco Herrera
University of Granada

José Manuel Benítez
University of Granada

Abstract

Fuzzy rule-based systems (FRBSs) are a well-known method family within soft computing. They are based on fuzzy concepts to address complex real-world problems. We present the R package **frbs** which implements the most widely used FRBS models, namely, Mamdani and Takagi Sugeno Kang (TSK) ones, as well as some common variants. In addition a host of learning methods for FRBSs, where the models are constructed from data, are implemented. In this way, accurate and interpretable systems can be built for data analysis and modeling tasks. In this paper, we also provide some examples on the usage of the package and a comparison with other common classification and regression methods available in R.

Keywords: fuzzy inference systems, soft computing, fuzzy sets, genetic fuzzy systems, fuzzy neural networks.

1. Introduction

This vignette is also published in the Journal of Statistical Software as Riza, Bergmeir, Herrera, and Benítez (2015). Please use this reference to cite this package. Fuzzy rule-based systems (FRBSs) are well known methods within soft computing, based on fuzzy concepts to address complex real-world problems. They have become a powerful method to tackle various problems such as uncertainty, imprecision, and non-linearity. They are commonly used for identification, classification, and regression tasks. FRBSs have been deployed in a number of engineering and science areas, e.g., in bioinformatics (Zhou, Lyons, Brophy, and Gravenor 2012), data mining (Ishibuchi, Nakashima, and Nii 2005a), control engineering (Babuska 1998), finance (Boyacioglu and Avci 2010), robotics (Bai, Zhuang, and Roth 2005), and pattern recognition (Chi, Yan, and Pham 1996). Furthermore, in addition to their effectiveness in practical applications, their acceptance grew strongly after they were proved to be universal approximators of continuous functions (Kosko 1992; Wang 1992).

FRBSs are also known as fuzzy inference systems or simply fuzzy systems. When applied to specific tasks, they also may receive specific names such as fuzzy associative memories or fuzzy controllers. They are based on the fuzzy set theory, proposed by Zadeh (1965), which aims at representing the knowledge of human experts in a set of fuzzy IF-THEN rules. Instead of us-

ing crisp sets as in classical rules, fuzzy rules use fuzzy sets. Rules were initially derived from human experts through knowledge engineering processes. However, this approach may not be feasible when facing complex tasks or when human experts are not available. An effective alternative is to generate the FRBS model automatically from data by using learning methods. Many methods have been proposed for this learning task such as space partition based methods (Wang and Mendel 1992), heuristic procedures (Ishibuchi, Nozaki, and Tanaka 1994), neural-fuzzy techniques (Jang 1993; Kim and Kasabov 1999), clustering methods (Chiu 1996; Kasabov and Song 2002), genetic algorithms (Cordon, Herrera, Hoffmann, and Magdalena 2001), gradient descent learning methods (Ichihashi and Watanabe 1990), etc.

On the Comprehensive R Archive Network (CRAN), there are already some packages present that make use of fuzzy concepts. The **sets** package (Meyer and Hornik 2009) includes the fundamental structure and operators of fuzzy sets: class construction, union, intersection, negation, etc. Additionally, it provides simple fuzzy inference mechanisms based on fuzzy variables and fuzzy rules, including fuzzification, inference, and defuzzification. The package **fuzzyFDR** (Lewin 2007) determines fuzzy decision rules for multiple testing of hypotheses with discrete data, and genetic algorithms for learning FRBSs are implemented in the package **fugeR** (Bujard 2012). The **e1071** package (Meyer, Dimitriadou, Hornik, Weingessel, and Leisch 2012) provides many useful functions for latent class analysis, support vector machines, etc. With respect to fuzzy concepts, this package offers implementations of algorithms for fuzzy clustering, and fuzzy k-means, which is an enhancement of the k-means clustering algorithm using fuzzy techniques.

The **frbs** package (Riza, Bergmeir, Herrera, and Benítez 2014), which we present in this paper, aims not only to provide the R community with all of the most prominent FRBS models but also to implement the most widely used learning procedures for FRBSs. Unlike the previous packages which implement FRBSs, we focus on learning from data with various learning methods such as clustering, space partitioning, neural networks, etc. Furthermore, we also provide the possibility to build FRBSs manually from expert knowledge. The package is available from the Comprehensive R Archive Network (CRAN) at `http://CRAN.R-project.org/package=frbs`.

The remainder of this paper is structured as follows. Section 2 gives an overview of fuzzy set theory and FRBSs. Section 3 presents the architecture and implementation details of the package. The usage of the package is explained in Section 4. In Section 5, we provide benchmarking experiments comparing package **frbs** against some other packages on CRAN from a simulation point of view. Then, in Section 6, the available packages on CRAN implementing fuzzy concepts are compared to package **frbs** in detail, based on their capabilities and functionalities. Finally, Section 7 concludes the paper.

2. Fuzzy rule-based systems

In this section, we provide a short overview of the theoretical background of the fuzzy set theory, FRBSs, and the associated learning procedures.

2.1. Overview of FRBSs

Fuzzy set theory was proposed by Zadeh (1965), as an extension of the classical set theory to model sets whose elements have degrees of membership. So, instead of just having two values: member or non-member, fuzzy sets allow for degrees of set membership, defined by a value between zero and one. A degree of one means that an object is a member of the set, a value of zero means it is not a member, and a value somewhere in-between shows a partial degree of membership. The grade of membership of a given element is defined by the so-called membership function. The theory proposes this new concept of a set, which is a generalization of the classic concept, and definitions for the corresponding operations, namely, union, intersection, complementary, and so forth. This in turn led to the extension of many other concepts, such as number, interval, equation, etc. Moreover, it happens that most fuzzy concepts come from concepts from human language, which is inherently vague. Fuzzy set theory provides the tools to effectively represent linguistic concepts, variables, and rules, becoming a natural model to represent human expert knowledge. A key concept is that of a linguistic variable, defined as a variable whose values are linguistic terms, each with a semantic described by a fuzzy set (Zadeh 1975). A linguistic value refers to a label for representing knowledge that has meaning determined by its degree of the membership function. For example, $a_1 =$ "*hot*" with the degree $\mu = 0.8$ means that the variable a_1 has a linguistic value represented by the label "*hot*", whose meaning is determined by the degree of 0.8.

During the last forty years, scientific research has been growing steadily and the available literature is vast. A lot of monographs provide comprehensive explanations about fuzzy theory and its techniques, for example in Klir and Yuan (1995); Pedrycz and Gomide (1998). One of the most fruitful developments of fuzzy set theory are FRBSs. We describe them in the following.

FRBSs are an extension of classical rule-based systems (also known as production systems or expert systems). Basically, they are expressed in the form "IF A THEN B" where A and B are fuzzy sets. A and B are called the antecedent and consequent parts of the rule, respectively. Let us assume we are trying to model the following problem: we need to determine the speed of a car considering some factors such as the number of vehicles in the street and the width of the street. So, let us consider three objects = {number of vehicles, width of street, speed of car} with linguistic values as follows:

Number of vehicles = {small, medium, large}.

Width of street = {narrow, medium, wide}.

Speed of car = {slow, medium, fast}.

Based on a particular condition, we can define a fuzzy IF-THEN rule as follows:

IF number of vehicles is small and width of street is medium **THEN** speed of car is fast.

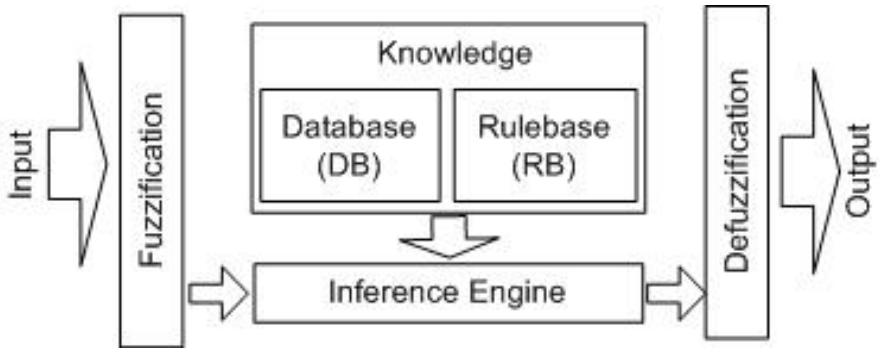

Figure 1: The components of the Mamdani model.

This example shows that rules using the fuzzy concept can be much easier to interpret and more flexible to change than classical rules. Indeed, the linguistic values are more understandable than the numerical form. With respect to the structure of the rule, there exist two basic FRBS models: the Mamdani and TSK models. The differences and characteristics of both models are discussed in the following.

The Mamdani model

This model type was introduced by Mamdani (1974) and Mamdani and Assilian (1975). It is built by linguistic variables in both the antecedent and consequent parts of the rules. So, considering multi-input and single-output (MISO) systems, fuzzy IF-THEN rules are of the following form:

$$\textbf{IF } X_1 \text{ is } A_1 \text{ and } \ldots \text{ and } X_n \text{ is } A_n \textbf{ THEN } Y \text{ is } B, \tag{1}$$

where X_i and Y are input and output linguistic variables, respectively, and A_i and B are linguistic values.

The standard architecture for the Mamdani model is displayed in Figure 1. It consists of four components: *fuzzification*, *knowledge base*, *inference engine*, and *defuzzifier*. The fuzzification interface transforms the crisp inputs into linguistic values. The knowledge base is composed of a database and a rulebase. While the database includes the fuzzy set definitions and parameters of the membership functions, the rulebase contains the collections of fuzzy IF-THEN rules. The inference engine performs the reasoning operations on the appropriate fuzzy rules and input data. The defuzzifier produces crisp values from the linguistic values as the final results.

Since the Mamdani model is built out of linguistic variables it is usually called a linguistic or descriptive system. A key advantage is that its interpretability and flexibility to formulate knowledge are higher than for other FRBSs. However, the model suffers some drawbacks. For example, its accuracy is lower for some complex problems, which is due to the structure of its linguistic rules (Cordon *et al.* 2001).

The TSK model

Instead of working with linguistic variables on the consequent part as in the Mamdani model in Equation 1, the TSK model (Takagi and Sugeno 1985; Sugeno and Kang 1988) uses rules

whose consequent parts are represented by a function of input variables. The most commonly used function is a linear combination of the input variables: $Y = f(X_1, \ldots, X_n)$ where X_i and Y are the input and output variables, respectively. The function $f(X_1, \ldots, X_n)$ is usually a polynomial in the input variables, so that we can express it as $Y = p_1 \cdot X_1 + \cdots + p_n \cdot X_n + p_0$ with a vector of real parameters $p = (p_0, p_1, \ldots, p_n)$. Since we have a function on the consequent part, the final output is a real value, so that there is no defuzzifier for the TSK model.

The TSK model has been successfully applied to a large variety of problems, particularly, when accuracy is a priority. Its success is mainly because this model type provides a set of system equations on the consequent parts whose parameters are easy to estimate by classical optimization methods. Their main drawback, however, is that the obtained rules are not so easy to interpret.

2.2. Variants of FRBSs

Other variants have been proposed in order to improve the accuracy and to handle specific problems. Their drawback is that they usually have higher complexity and are less interpretable. For example, the disjunctive normal form (DNF) fuzzy rule type has been used in González, Pérez, and Verdegay (1993). It improves the Mamdani model in Equation 1 on the antecedent part, in the sense that the objects are allowed to consider more than one linguistic value at a time. These linguistic values are joined by a disjunctive operator. The approximate Mamdani type proposed by Herrera, Lozano, and Verdegay (1998) may have a different set of linguistic values for each rule instead of sharing a common definition of linguistic values as it is the case of the original Mamdani formulation. So they are usually depicted by providing the values of the corresponding membership function parameters instead of a linguistic label. The advantages of this type are the augmented degree of freedom of parameters so that for a given number of rules the system can better be adapted to the complexity of the problems. Additionally, the learning processes can identify the structure and estimate the parameters of the model at the same time.

Fuzzy rule-based classification systems (FRBCS) are specialized FRBSs to handle classification tasks. A main characteristic of classification is that the outputs are categorical data. Therefore, in this model type we preserve the antecedent part of linguistic variables, and change the consequent part to be a class C_j from a prespecified class set $C = \{C_1, \ldots, C_M\}$. Three structures of fuzzy rules for classification tasks can be defined as follows. The simplest form introduced by Chi *et al.* (1996) is constructed with a class in the consequent part. The FRBCS model with a certainty degree (called weight) in the consequent part was discussed in Ishibuchi, Nozaki, and Tanaka (1992). FRBCS with a certainty degree for all classes in the consequent part are proposed by Mandal, Murthy, and Pal (1992). It means that instead of considering one class, this model provides prespecified classes with their respective weights for each rule.

2.3. Constructing FRBSs

Constructing an FRBS means defining all of its components, especially the database and rulebase of the knowledge base. The operator set for the inference engine is selected based on the application or kind of model. For example, minimum or product are common choices for the conjunction operator. But the part that requires the highest effort is the knowledge base.

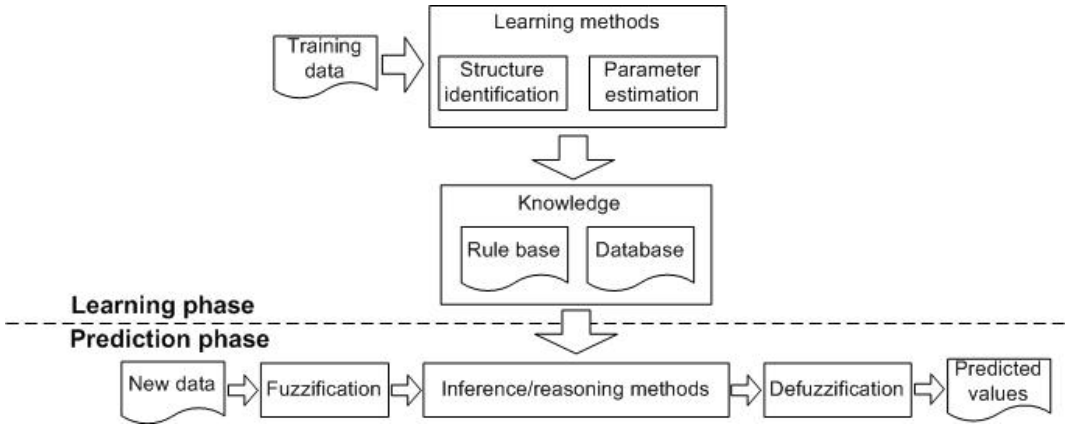

Figure 2: Learning and prediction phase of an FRBS.

Basically, there are two different strategies to build FRBSs, depending on the information available (Wang 1994). The first strategy is to get information from human experts. It means that the knowledge of the FRBS is defined manually by knowledge engineers, who interview human experts to extract and represent their knowledge. However, there are many cases in which this approach is not feasible, e.g., experts are not available, there is not enough knowledge available, etc. The second strategy is to obtain FRBSs by extracting knowledge from data by using learning methods. In the **frbs** package a host of learning methods for FRBS building is implemented.

Generally the learning process involves two steps: structure identification and parameter estimation (Sugeno and Yasukawa 1993; Pedrycz 1996). In the structure identification step, we determine a rulebase corresponding to pairs of input and output variables, and optimize the structure and number of the rules. Then, the parameters of the membership function are optimized in the parameter estimation step. The processing steps can be performed sequentially or simultaneously.

Regarding the components of the FRBSs that need to be learned or optimized, the following has to be performed:

- Rulebase: Qualified antecedent and consequent parts of the rules need to be obtained, the number of rules needs to be determined and the rules have to be optimized.
- Database: Optimized parameters of the membership functions have to be defined.
- Weight of rules: Especially for fuzzy rule-based classification systems, optimized weights of each rule have to be calculated.

After the inference engine operators are set and the knowledge base is built, the FRBS is ready. Obviously, as in other modeling or machine learning methods, a final validation step is required. After achieving a successful validation the FRBS is ready for use. Figure 2 shows the learning and prediction stages of an FRBS. An FRBS can be used just like other classification or regression models – e.g., classification trees, artificial neural networks, Bayesian networks, ..., – and a leading design goal when approaching the development of the package **frbs** was endowing it with an interface as similar as possible to implementations in R of such models.

3. Package architecture and implementation details

The **frbs** package is written in pure R using S3 classes and methods. It provides more than ten different learning methods in order to construct FRBSs for regression and classification tasks from data. These methods are listed in Table 1. The main interface of the package is shown in Table 2. The `frbs.learn()` function and the `predict()` method for '`frbs`' objects are used to construct FRBS models and perform fuzzy reasoning, respectively. Figure 3 shows the internal functions of the different method implementations which are invoked through `frbs.learn()`.

Method name	Description	FRBS model	Grouping	Tasks
`"ANFIS"`	Adaptive-network-based fuzzy inference system	`TSK`	Fuzzy neural networks	Regression
`"DENFIS"`	Dynamic evolving neural-fuzzy inference system	`CLUSTERING`	Clustering	Regression
`"FH.GBML"`	Ishibuchi's method based on hybridization of `"GFS.GCCL"` and the Pittsburgh approach	`FRBCS`	Genetic fuzzy systems	Classification
`"FIR.DM"`	Fuzzy inference rules by descent method	`TSK`	Gradient descent	Regression
`"FRBCS.CHI"`	FRBCS based on Chi's technique	`FRBCS`	Space partition	Classification
`"FRBCS.W"`	FRBCS based on Ishibuchi's technique using weight factor	`FRBCS`	Space partition	Classification
`"FS.HGD"`	FRBS using heuristics and gradient descent method	`TSK`	Gradient descent	Regression
`"GFS.FR.MOGUL"`	Genetic fuzzy for fuzzy rule learning based on the MOGUL methodology	`APPROXIMATE`	Genetic fuzzy systems	Regression
`"GFS.GCCL"`	Ishibuchi's method based on genetic cooperative-competitive learning	`FRBCS`	Genetic fuzzy systems	Classification
`"GFS.THRIFT"`	Genetic fuzzy system based on Thrift's method	`MAMDANI`	Genetic fuzzy systems	Regression
`"HYFIS"`	Hybrid neural fuzzy inference system	`MAMDANI`	Fuzzy neural networks	Regression
`"SBC"`	Subtractive clustering	`CLUSTERING`	Clustering	Regression
`"SLAVE"`	Structural learning algorithm on vague environment	`FRBCS`	Genetic fuzzy systems	Classification
`"WM"`	Wang and Mendel's technique	`MAMDANI`	Space partition	Regression

Table 1: The learning methods implemented in the **frbs** package.

Main functions	Description
`frbs.learn()`	The main function of the package to construct an '`frbs`' object automatically from data.
S3 method `predict()`	This function performs fuzzy reasoning to obtain predicted values for new data, using a given '`frbs`' object.
`frbs.gen()`	This function can be used to construct the '`frbs`' object manually from expert knowledge.
S3 method `summary()`	Show a summary of an '`frbs`' object.
`plotMF()`	Plot the membership function.

Table 2: The main functions of the package.

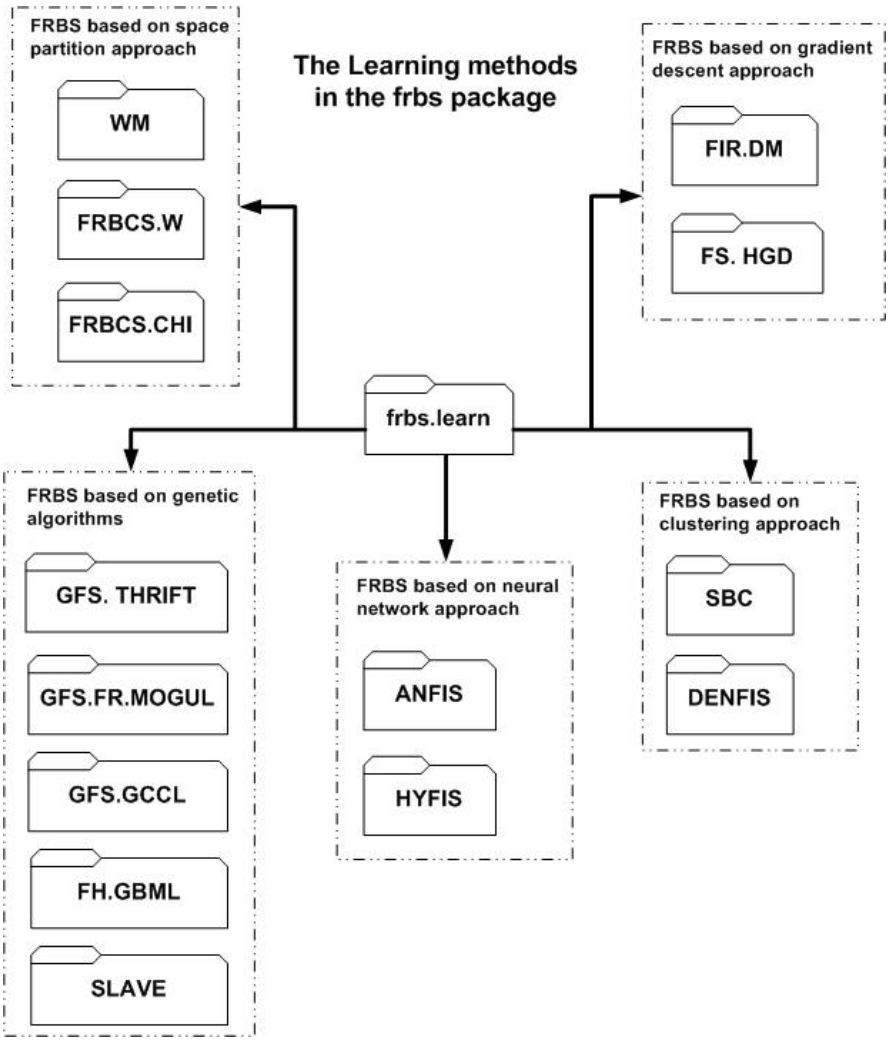

Figure 3: Functions for learning in the **frbs** package.

We classify the learning methods into five groups: FRBSs based on space partition, genetic algorithms, clustering, neural networks, and gradient descent. In the following, we discuss these five groups in detail.

3.1. FRBSs based on space partition approaches

Learning methods included in this group use a strategy of dividing the variable space, and then considering this partition to obtain the parameters of the membership functions. The following methods use space partition approaches to build FRBSs.

Wang and Mendel's technique (`"WM"`). It was proposed by Wang and Mendel (1992) using the Mamdani model. For the learning process, there are four stages as follows:

Step 1: Divide equally the input and output spaces of the given numerical data into fuzzy regions as the database. In this case, fuzzy regions refer to intervals for the linguistic terms. Therefore, the length of the fuzzy regions is related to the number of linguistic terms. For example, let us assume a concept of temperature between 1 and 5. Then, we define the linguistic terms "cold", "neutral", and "hot", and we define the length of fuzzy regions as 2. This now gives us the fuzzy regions as intervals $[1, 3]$, $[2, 4]$, $[3, 5]$, respectively, and we can construct triangular membership functions. E.g., in the first case, we have the corner points $a = 1$, $b = 2$, and $c = 3$ where b is a middle point whose degree of the membership function equals one.

Step 2: Generate fuzzy IF-THEN rules covering the training data, using the database from Step 1. First, we calculate degrees of the membership function for all values in the training data. For each instance and each variable, a linguistic value is determined as the linguistic term whose membership function is maximal in this case. Then, we repeat the process for all instances in the training data to construct fuzzy rules covering the training data.

Step 3: Determine a degree for each rule. Degrees of each rule are determined by aggregating degrees of membership functions in the antecedent and consequent parts. In this case, we are using the product aggregation operators.

Step 4: Obtain a final rulebase after deleting redundant rules. Considering the degrees of rules, we can delete a redundant rule with a lower degree.

The outcome is a Mamdani model.

FRBCS using Chi's method (`"FRBCS.CHI"`). This method was proposed by Chi *et al.* (1996), which is an extension of Wang and Mendel's method, for tackling classification problems. Basically, the algorithm is quite similar to Wang and Mendel's technique. Since it is based on the FRBCS model, Chi's method only takes class labels from the data to be consequent parts of fuzzy IF-THEN rules. In other words, we generate rules as in Wang and Mendel's technique and then we replace consequent parts with classes. Regarding calculation of degrees of each rule, they are determined by the antecedent part of the rules. Redundant rules can be deleted by considering their degrees. Lastly, we obtain fuzzy IF-THEN rules based on the FRBCS model.

FRBCS using Ishibuchi's method with weight factor (`"FRBCS.W"`). This method is adopted from Ishibuchi and Nakashima (2001). It implements the second type of FRBCS

which has certainty grades (weights) in the consequent parts of the rules. The antecedent parts are then determined by a grid-type fuzzy partition from the training data. The consequent class is defined as the dominant class in the fuzzy subspace corresponding to the antecedent part of each fuzzy IF-THEN rule. The class of a new instance is determined by the consequent class of the rule with the maximum product of its compatibility and certainty grades. The compatibility grade is determined by aggregating degrees of the membership function of antecedent parts while the certainty grade is calculated from the ratio among the consequent class.

3.2. FRBSs based on neural networks

The systems in this group are commonly also called neuro-fuzzy systems or fuzzy neural networks (FNN; Buckley and Hayashi 1994) since they combine artificial neural networks (ANN) with FRBSs. An FRBS is laid upon the structure of an ANN and the learning algorithm of the latter is used to adapt the FRBS parameters, usually the membership function parameters. There exist many variants of methods based on FNNs, such as the adaptive-network-based fuzzy inference system (`"ANFIS"`) and the hybrid neural fuzzy inference system (`"HYFIS"`). Both methods are implemented in the **frbs** package.

Adaptive-network-based fuzzy inference system (`"ANFIS"`). This method was proposed by Jang (1993). It considers a TSK FRBS model which is built out of a five-layered network architecture. The `"ANFIS"` learning algorithm consists of two processes, the forward and the backward stage. The forward stage goes through the five layers as follows:

Layer 1: The fuzzification process which transforms crisp into linguistic values using the Gaussian function as the shape of the membership function.

Layer 2: The inference stage using the t-norm operator (the AND operator).

Layer 3: Calculating the ratio of the strengths of the rules.

Layer 4: Calculating the parameters for the consequent parts.

Layer 5: Calculating the overall output as the sum of all incoming signals.

The backward stage is a process to estimate the database which consists of the parameters of the membership functions in the antecedent part and the coefficients of the linear equations in the consequent part. Since this method uses the Gaussian function as membership function, we optimize two parameters of this function: mean and variance. In this step, the least squares method is used to perform the parameter learning. For the prediction phase, the method performs normal fuzzy reasoning of the TSK model.

Hybrid neural fuzzy inference system (`"HYFIS"`). This learning procedure was proposed by Kim and Kasabov (1999). It uses the Mamdani model as its rule structure. There are two phases in this method for learning, namely the knowledge acquisition module and the structure and parameter learning. The knowledge acquisition module uses the techniques of Wang and Mendel. The learning of structure and parameters is a supervised learning method using gradient descent-based learning algorithms. The function generates a model

that consists of a rule database and parameters of the membership functions. `"HYFIS"` uses the Gaussian function as the membership function. So, there are two parameters which are optimized: mean and variance of the Gaussian function for both antecedent and consequent parts. Predictions can be performed by the standard Mamdani procedure.

3.3. FRBSs based on genetic algorithms

Genetic fuzzy systems (GFS; Cordon *et al.* 2001) are a combination of genetic algorithms and FRBSs. Generally, the genetic algorithms are used to search and optimize the parameters of the membership functions and of the fuzzy rule construction process. The following methods have been implemented in the **frbs** package.

Genetic fuzzy system based on Thrift's method (`"GFS.THRIFT"`). Thrift (1991) introduces a technique for learning of Mamdani models based on a genetic algorithm. In this method, we build a population from the available options of the rules. Each rule represents one chromosome. A new population is obtained through standard crossover and mutation operators applied to the chromosomes. The fitness of an individual is determined as the root mean square error (RMSE) between the actual and predicted values. The predicted values are obtained from fuzzy reasoning using the Mamdani model as described in Section 2.3. The final solution is obtained as the best individual after generating the maximal number of generations. The method tries to find the best configuration of the rulebase without changing the database.

Genetic fuzzy systems for fuzzy rule learning based on the MOGUL methodology (`"GFS.FR.MOGUL"`). This method is proposed by Herrera *et al.* (1998). It uses a genetic algorithm to determine the structure of the fuzzy rules and the parameters of the membership functions simultaneously. To achieve this, it uses the approximative approach as mentioned in Section 2.2. Each fuzzy rule is modeled as a chromosome which consists of the parameter values of the membership function. So, every rule has its own membership function values. A population contains many such generated chromosomes, based on the iterative rule learning approach (IRL). IRL means that the best chromosomes will be generated one by one according to the fitness value and covering factor. The method carries out the following steps:

Step 1: Genetic generation process involving the following steps: Create an initial population, evaluate individual fitness, perform genetic operators, obtain the best rule and collect it, and repeat this process until the stopping criterion has been met.

Step 2: Tuning process: Repetitively adjust the best individual until the stopping criterion is met.

Step 3: Obtain an FRBS model as the output.

Ishibuchi's method based on genetic cooperative competitive learning (`"GFS.GCCL"`). This method is based on Ishibuchi, Nakashima, and Murata (1999) using genetic cooperative competitive learning (GCCL) to handle classification problems. In this method, a chromosome describes each linguistic IF-THEN rule using integers as its representation of the antecedent

part. In the consequent part of the fuzzy rules, the heuristic method is carried out to automatically generate the class. The evaluation is calculated for each rule, which means that the performance is not based on the entire rule set. The method works as follows:

Step 1: Generate an initial population of fuzzy rules.

Step 2: Evaluate each fuzzy rule in the current population.

Step 3: Generate new fuzzy rules by genetic operators.

Step 4: Replace a part of the current population with the newly generated rules.

Step 5: Terminate the algorithm if the stopping condition is satisfied, otherwise return to Step 2.

Additionally, to handle high-dimensional data, this method proposes "don't care" attributes in the antecedent fuzzy sets. This means that linguistic values which have "don't care" are always assumed to have a degree of one.

Ishibuchi's method based on hybridization of GCCL and Pittsburgh (`"FH.GBML"`). This method is based on Ishibuchi's method using the hybridization of GCCL and Pittsburgh approach for GFSs (Ishibuchi, Yamamoto, and Nakashima 2005b). The algorithm of this method is as follows:

Step 1: Generate a population where each individual of the population is a fuzzy rule set.

Step 2: Calculate the fitness value of each rule set in the current population.

Step 3: Generate new rule sets by selection, crossover, and mutation in the same manner as in the Pittsburgh-style algorithm. Then, apply iterations of the GCCL-style algorithm to each of the generated rule sets by considering user-defined probabilities of crossover and mutation.

Step 4: Add the best rule set in the current population to newly generated rule sets to form the next population.

Step 5: Return to Step 2 if the prespecified stopping condition is not satisfied.

Structural learning algorithm on vague environment (`"SLAVE"`). This method is adopted from Gonzalez and Peréz (2001). `"SLAVE"` is based on the IRL approach which means that we get only one fuzzy rule in each execution of the genetic algorithm. To eliminate the irrelevant variables in a rule, `"SLAVE"` has a structure composed of two parts: the first part is to represent the relevance of variables and the second one is to define values of the parameters. The following steps are conducted in order to obtain fuzzy rules:

Step 1: Use a genetic algorithm process to obtain *one rule* for the FRBS.

Step 2: Collect the rule into the final set of rules.

Step 3: Check and penalize this rule.

Step 4: If the stopping criterion is satisfied, the system returns the set of rules as solution. Otherwise, go back to Step 1.

This method applies binary codes as representation of the population and conducts the basic genetic operators, i.e., selection, crossover, and mutation on the population. Then, the best rule is determined as the rule with the highest consistency and completeness degree.

3.4. FRBSs based on clustering approaches

Fuzzy rules can be constructed by clustering approaches through representing cluster centers as rules. Two strategies to obtain cluster centers are implemented in the **frbs** package as follows.

Subtractive clustering (`"SBC"`). This method is proposed by Chiu (1996). For generating the rules in the learning phase, the `"SBC"` method is used to obtain the cluster centers. It is an extension of Yager and Filev's mountain method (Yager and Filev 1994). It considers each data point as a potential cluster center by determining the potential of a data point as a function of its distances to all the other data points. A data point has a high potential value if that data point has many nearby neighbors. The highest potential is chosen as the cluster center and then the potential of each data point is updated. The process of determining new clusters and updating potentials repeats until the remaining potential of all data points falls below some fraction of the potential of the first cluster center. After getting all the cluster centers from `"SBC"`, the cluster centers are optimized by fuzzy c-means.

Dynamic evolving neural fuzzy inference system (`"DENFIS"`). This method is proposed by Kasabov and Song (2002). There are several steps in this method that are to determine the cluster centers using the evolving clustering method (`ECM`), to partition the input space and to find optimal parameters for the consequent part of the TSK model, using a least squares estimator. The `ECM` algorithm is a distance-based clustering method which is determined by a threshold value, `Dthr`. This parameter influences how many clusters are created. In the beginning of the clustering process, the first instance from the training data is chosen to be a cluster center, and the determining radius is set to zero. Afterwards, using the next instance, cluster centers and radius are changed based on certain mechanisms of `ECM`. All of the cluster centers are then obtained after evaluating all the training data. The next step is to update the parameters on the consequent part with the assumption that the antecedent part that we got from `ECM` is fixed. Actually, `ECM` can perform well as an online clustering method, but here it is used in an offline mode.

3.5. FRBSs based on the gradient descent approach

Some methods use a gradient descent approach to optimize the parameters on both antecedent and consequent parts of the rules. The following methods of this family are implemented in the package.

Fuzzy inference rules with descent method (`"FIR.DM"`). This method is proposed by Nomura, Hayashi, and Wakami (1992). `"FIR.DM"` uses simplified fuzzy reasoning where

the consequent part is a real number (a particular case within the TSK model), while the membership function on the antecedent part is expressed by an isosceles triangle. So, in the learning phase, `"FIR.DM"` updates three parameters which are center and width of the triangle and a real number on the consequent part using a descent method.

FRBS using heuristics and the gradient descent method (`"FS.HGD"`). This method is proposed by Ishibuchi *et al.* (1994). It uses fuzzy rules with non-fuzzy singletons (i.e., real numbers) in the consequent parts. The techniques of space partitioning are implemented to generate the antecedent part, while the initial consequent part of each rule is determined by the weighted mean value of the given training data. Then, the gradient descent method updates the value of the consequent part. Furthermore, the heuristic value given by the user affects the value of weight of each data point.

4. Using the frbs package

In this section, we discuss the usage of the package. We show how to generate FRBSs from data and predict using new data. Basically, the following steps are performed to use a learning method from the package. Firstly, in a preprocessing step the data and parameters need to be prepared. Then, the `frbs.learn()` function is used to generate the FRBS. The `summary()` function can then be used to show the FRBS. We note that the FRBS can contain different components depending on the method which was used to build it. To display the shape of the membership functions, we use then the function `plotMF()`. Finally, prediction with new values is performed by calling `predict()`.

In the following example[1], we demonstrate the use of a particular learning method in package **frbs** which is an FRBCS model with weight factor (`"FRBCS.W"`) for handling a classification problem. Using other methods from the package is very similar. In this example, we consider the `iris` dataset which is a popular benchmarking dataset for classification problems.

4.1. Preprocessing

Generally, there are four inputs/parameters needed for all learning methods implemented in the **frbs** package, which are the data, the range of the data (can be omitted), the method type and the control parameters array. Firstly, the data must be a data frame or matrix $(m \times n)$ where m is the number of instances and n is the number of variables; the last column is the output variable. It should be noted that the training data must be expressed in numbers (numerical data). In experiments, we usually divide the data into two groups: training and testing data. The data ranges need to be compiled into a matrix $(2 \times n)$ where the first and second rows are minimum and maximum values, respectively, and n is the number of input variables. If the ranges are not given by the user, they are automatically computed from the data. The `iris` dataset is available directly in R. We convert the categorical target values into numerical data and split the data into training and test sets with the following:

```
R> data("iris", package = "datasets")
R> irisShuffled <- iris[sample(nrow(iris)), ]
```

[1] Due to space constraints only one example is shown here. Further examples and more detailed information on the package usage can be found at http://dicits.ugr.es/software/FRBS/.

```
R> irisShuffled[, 5] <- unclass(irisShuffled[, 5])
R> range.data.input <- apply(iris[, -ncol(iris)], 2, range)
R> tra.iris <- irisShuffled[1:140, ]
R> tst.iris <- irisShuffled[141:nrow(irisShuffled), 1:4]
R> real.iris <- matrix(irisShuffled[141:nrow(irisShuffled), 5], ncol = 1)
```

4.2. Model generation

To generate the FRBS, we use the function `frbs.learn()`. It has some parameters that need to be set. The `method.type` is the name of the learning method to use, and a list of method-dependent parameters needs to be supplied in the `control` argument. In this example, we use the FRBCS model with weight factor based on Ishibuchi's technique. So we assign the `method.type` to the value `"FRBCS.W"`. A list of the name of all implemented methods can be found in Table 1. In the `"FRBCS.W"` method, there are two arguments we need to set in the `control` argument: the number of linguistic terms (`num.labels`) and the shape of the membership function (`type.mf`). If we do not set them, package **frbs** will use default options. In this example, we choose the number of linguistic terms to be 3 and the shape of the membership function to be `"TRAPEZOID"`. Common values for the number of linguistic terms are 3, 5, 7, or 9 though this depends on the complexity of the problem under consideration and the desired level of accuracy. However, a higher number of terms makes the FRBS more difficult to be interpreted. We generate the model as follows:

```
R> object.frbcs.w <- frbs.learn(tra.iris, range.data.input,
+    method.type = "FRBCS.W", control = list(num.labels = 3,
+    type.mf = "TRAPEZOID"))
```

After generating the FRBS, we can display its characteristics by executing `summary()`.

```
R> summary(object.frbcs.w)

The name of model:  sim-0
Model was trained using:  FRBCS.W
The names of attributes:  Sepal.Length Sepal.Width Petal.Length Petal.Width
Species
The interval of input data:
    Sepal.Length Sepal.Width Petal.Length Petal.Width
min          4.3         2.0          1.0         0.1
max          7.9         4.4          6.9         2.5
Type of FRBS model:
[1] "FRBCS"
Type of membership functions:
[1] "TRAPEZOID"
Type of t-norm method:
[1] "Standard t-norm (min)"
Type of s-norm method:
[1] "Standard s-norm"
Type of implication function:
```

```
[1] "ZADEH"
The names of linguistic terms on the input variables:
 [1] "small"  "medium" "large"  "small"  "medium" "large"  "small"
 [8] "medium" "large"  "small"  "medium" "large"
The parameter values of membership function on the input variable
(normalized):
     small medium large small medium large small medium large small
[1,]   2.0   4.00   3.0   2.0   4.00   3.0   2.0   4.00   3.0   2.0
[2,]   0.0   0.23   0.6   0.0   0.23   0.6   0.0   0.23   0.6   0.0
[3,]   0.2   0.43   0.8   0.2   0.43   0.8   0.2   0.43   0.8   0.2
[4,]   0.4   0.53   1.0   0.4   0.53   1.0   0.4   0.53   1.0   0.4
[5,]    NA   0.73    NA    NA   0.73    NA    NA   0.73    NA    NA
     medium large
[1,]   4.00   3.0
[2,]   0.23   0.6
[3,]   0.43   0.8
[4,]   0.53   1.0
[5,]   0.73    NA
The number of linguistic terms on each variables
     Sepal.Length Sepal.Width Petal.Length Petal.Width Species
[1,]            3           3            3           3       3
The fuzzy IF-THEN rules:
   V1           V2 V3     V4  V5          V6 V7     V8  V9
1  IF Sepal.Length is  small and Sepal.Width is medium and
2  IF Sepal.Length is  small and Sepal.Width is  small and
3  IF Sepal.Length is  large and Sepal.Width is medium and
4  IF Sepal.Length is medium and Sepal.Width is  large and
5  IF Sepal.Length is medium and Sepal.Width is medium and
6  IF Sepal.Length is medium and Sepal.Width is medium and
7  IF Sepal.Length is medium and Sepal.Width is  small and
8  IF Sepal.Length is medium and Sepal.Width is  small and
9  IF Sepal.Length is  large and Sepal.Width is  small and
10 IF Sepal.Length is  small and Sepal.Width is  large and
11 IF Sepal.Length is  large and Sepal.Width is  large and
12 IF Sepal.Length is  small and Sepal.Width is  small and
13 IF Sepal.Length is medium and Sepal.Width is  small and
14 IF Sepal.Length is  large and Sepal.Width is medium and
15 IF Sepal.Length is  large and Sepal.Width is medium and
16 IF Sepal.Length is medium and Sepal.Width is medium and
17 IF Sepal.Length is medium and Sepal.Width is medium and
18 IF Sepal.Length is  small and Sepal.Width is medium and
19 IF Sepal.Length is  large and Sepal.Width is medium and
20 IF Sepal.Length is medium and Sepal.Width is  small and
21 IF Sepal.Length is medium and Sepal.Width is medium and
22 IF Sepal.Length is  small and Sepal.Width is  small and
23 IF Sepal.Length is medium and Sepal.Width is  small and
            V10 V11    V12 V13         V14 V15    V16  V17     V18
```

```
1  Petal.Length  is  small and Petal.Width  is  small THEN Species
2  Petal.Length  is  small and Petal.Width  is  small THEN Species
3  Petal.Length  is  large and Petal.Width  is  large THEN Species
4  Petal.Length  is  small and Petal.Width  is  small THEN Species
5  Petal.Length  is  large and Petal.Width  is  large THEN Species
6  Petal.Length  is medium and Petal.Width  is medium THEN Species
7  Petal.Length  is medium and Petal.Width  is medium THEN Species
8  Petal.Length  is  large and Petal.Width  is medium THEN Species
9  Petal.Length  is  large and Petal.Width  is  large THEN Species
10 Petal.Length  is  small and Petal.Width  is  small THEN Species
11 Petal.Length  is  large and Petal.Width  is  large THEN Species
12 Petal.Length  is medium and Petal.Width  is medium THEN Species
13 Petal.Length  is  large and Petal.Width  is  large THEN Species
14 Petal.Length  is  large and Petal.Width  is medium THEN Species
15 Petal.Length  is medium and Petal.Width  is medium THEN Species
16 Petal.Length  is medium and Petal.Width  is  large THEN Species
17 Petal.Length  is medium and Petal.Width  is  large THEN Species
18 Petal.Length  is medium and Petal.Width  is medium THEN Species
19 Petal.Length  is  large and Petal.Width  is  large THEN Species
20 Petal.Length  is medium and Petal.Width  is  large THEN Species
21 Petal.Length  is  large and Petal.Width  is medium THEN Species
22 Petal.Length  is medium and Petal.Width  is  large THEN Species
23 Petal.Length  is  large and Petal.Width  is medium THEN Species
   V19 V20
1   is   1
2   is   1
3   is   3
4   is   1
5   is   3
6   is   2
7   is   2
8   is   3
9   is   3
10  is   1
11  is   3
12  is   2
13  is   3
14  is   3
15  is   2
16  is   3
17  is   2
18  is   2
19  is   2
20  is   3
21  is   3
22  is   3
23  is   2
```

```
The certainty factor:

 [1,] 0.3729181
 [2,] 0.3729181
 [3,] 0.6702364
 [4,] 0.3729181
 [5,] 0.6702364
 [6,] 0.4568455
 [7,] 0.4568455
 [8,] 0.6702364
 [9,] 0.6702364
[10,] 0.3729181
[11,] 0.6702364
[12,] 0.4568455
[13,] 0.6702364
[14,] 0.6702364
[15,] 0.4568455
[16,] 0.6702364
[17,] 0.4568455
[18,] 0.4568455
[19,] 0.4568455
[20,] 0.6702364
[21,] 0.6702364
[22,] 0.6702364
[23,] 0.4568455
```

In this case, the FRBS consists of the following elements:

Name of model: Model name given by the user.

Model was trained using: The learning method that was used for model building.

Interval of training data: The range data of the original training data.

Number of linguistic terms on the input variables: In this example, we use 3 linguistic terms for the input variables.

Names of linguistic terms on the input variables: These names are generated automatically by package **frbs** expressing all linguistic terms considered. Generally, the names are built with two parts which are the name of the variable expressed by `"v"` and the name of the linguistic label of each variable represented by `"a"`. For example, `"v.1_a.1"` means the linguistic label `a.1` of the first variable. However, we provide different formats when we set the `num.labels` parameter to 3, 5, and 7. In this example, `num.labels` is 3 and linguistic terms are `"small"`, `"medium"`, and `"large"` for each input variables.

Parameter values of membership function on input variables (normalized): A matrix $(5 \times n)$ where n depends on the number of linguistic terms of the input variables and the first row of the matrix describes the type of the membership function, and the rest of the rows express the parameter values. Additionally, the column expresses the

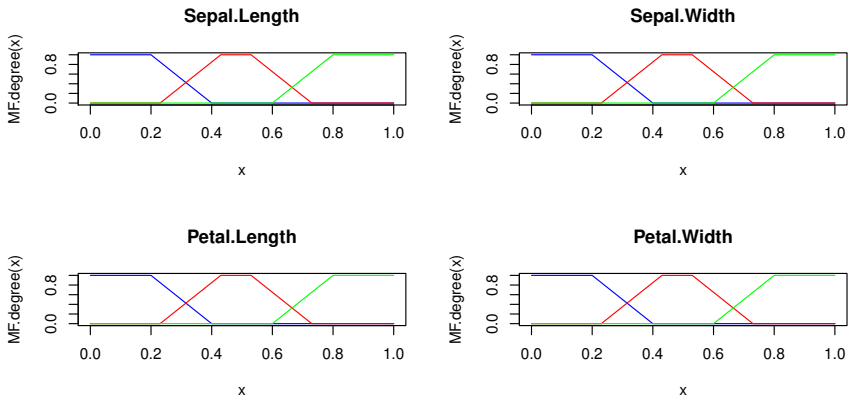

Figure 4: The plot of membership functions.

linguistic terms which are shown as column names. For example, the label `"medium"` has a value of {4.0, 0.23, 0.43, 0.53, 0.73}, where 4.0 is an indicator of trapezoid shape, and 0.23, 0.43, 0.53, and 0.73 are corner points of the trapezoid. The complete list of all shapes can be found in the **frbs** manual.

Fuzzy IF-THEN rules: They represent the knowledge base containing two parts: antecedent and consequent parts which are separated by "THEN". In this case, we use a fuzzy rule-based classification system so we have a predefined class on the consequent part. Note that the form of rules may be different, depending on the model used.

Weight of the rules: Since we use the `"FRBCS.W"` method, every rule has a corresponding weight in this matrix.

The above description can be different when using other methods. The complete components are explained in the **frbs** manual.

Furthermore, the plot of membership functions can be seen in Figure 4. It shows that there are four input attributes which have three linguistic terms for each attribute. The range of data has been normalized to lie between zero and one along the horizontal axis, and the vertical axis presents the degree of the membership function.

4.3. Prediction with new data

Prediction with new data is quite simple, and familiar to R users by using the `predict()` function with the FRBS model (namely, `object.frbcs.w`) and the new data (in this case, we use `tst.iris` as new data) as its arguments:

```
R> pred <- predict(object.frbcs.w, tst.iris)
```

After prediction, we can, for example, calculate the error percentage to know the accuracy of the model as follows:

```
R> err <- 100 * sum(pred != real.iris) / length(pred)
R> err

[1] 0
```

In this example, the test set of the `iris` dataset is classified entirely correctly.

5. Experimental study: Comparison with other packages

This section presents an experimental comparison of the **frbs** package with other packages available on CRAN. The goal of this comparison is basically to illustrate that the performance of FRBSs is competitive to other approaches. More detailed studies of the merits of FRBSs can be found in the specialized literature (e.g., Klir and Yuan (1995); Pedrycz and Gomide (1998), (Babuska 1998), (Boyacioglu and Avci 2010)). The comparison includes both regression and classification problems. In regression, the response or output variable is numerical/continuous, whereas in classification the output is a category. We perform experiments using several datasets to evaluate the methods.

5.1. Regression tasks

In the following, we describe the experiment design for regression, which includes the datasets, the methods considered for comparison, their parameters, and finally the experimental results.

Datasets

In this task, we consider two time series, which are the gas furnace dataset (Box and Jenkins 1970) and the Mackey-Glass series (Mackey and Glass 1977). Both datasets have been included in the package. Originally, the first dataset has two attributes which are methane gas and the percentage of carbon dioxide inside the gas furnace. However, in this experiment we arrange the dataset as follows. As input variables, we use 292 consecutive values of methane at time $(t-4)$ and the CO_2 at time $(t-1)$, with the produced CO_2 at time (t) as an output variable. In other words, each training data point consists of $[u(t-4), y(t-1), y(t)]$, where u is methane and y is CO_2. Then, we divide the data into two groups: 70% of the data are used for training and the rest of the data is used for testing.

Secondly, the Mackey-Glass chaotic time series is defined by the following delayed differential equation:

$$\frac{dx(t)}{dt} = \frac{(\alpha \times x(t-\tau)}{(1 + x(t-\tau)^{10}))} - \beta \times x(t)$$

Using the above equation, we generate 1000 samples, with input parameters as follows: $\alpha = 0.2$, $\beta = 0.1$, $\tau = 17$, $x_0 = 1.2$, $dt = 1$. The dataset is embedded in the following way: input variables: $x(t-18)$, $x(t-12)$, $x(t-6)$, $x(t)$ and output variable: $x(t+6)$. After that, we split the data into two equal sized datasets (i.e., training and testing data).

Methods considered for comparison and their parameters

The following R packages are used to compare them to the **frbs** package. The selection of packages is certainly not exhaustive, however, it is a representative set of well-known standard methods, which furthermore have different characteristics and implement different approaches to deal with the regression problems.

- **randomForest** (**?**Breiman, Cutler, Liaw, and Wiener 2012): This package implements the random forests method, which combines various decision trees to predict or classify

Methods	Parameters
`randomForest`	`importance = TRUE, proximity = TRUE`
`mlp`	`size = 5, learnFuncParams =` $[0, 1]$, `maxit = 350`
`lm`	none
`glm`	none
`ppr`	none
`nnet`	`size = 30, linout = TRUE, maxit = 1000`
`regTree`	none
`svm`	`cost = 10, gamma = 0.01`
`"ANFIS"`	`num.labels = 5, max.iter = 300, step.size = 0.01, type.mf = 3`
`"HYFIS"`	`num.labels = 5, max.iter = 200, step.size = 0.01`
`"SBC"`	`r.a = 0.3, eps.high = 0.5, eps.low = 0.15`
`"DENFIS"`	`Dthr = 0.15, max.iter = 5000, step.size = 0.01, d = 2`
`"FIR.DM"`	`num.labels = 5, max.iter = 1000, step.size = 0.01`
`"FS.HGD"`	`num.labels = 5, max.iter = 100, step.size = 0.01, alpha.heuristic = 1`
`"GFS.THRIFT"`	`popu.size = 30, num.labels = 5, persen_cross = 0.9, persen_mutant = 0.3, max.gen = 100`
`"GFS.FR.MOGUL"`	`persen_cross = 0.9, max.iter = 300, max.gen = 200, max.tune = 500, persen_mutant = 0.3, epsilon = 0.95`
`"WM"`	`num.labels = 15, type.mf = 3, type.defuz = 1, type.tnorm = 1, type.snorm = 1`

Table 3: The parameters of the methods selected for comparison for regression.

the values. The method is suitable for both classification and regression tasks.

- **RSNNS** (Bergmeir and Benítez 2012): This package implements many standard procedures of neural networks. Here, we use the multi-layer perceptron (`mlp`) for regression.

- **fRegression** (Wuerts and et al. 2012): This package implements various methods for regression tasks. We use three methods from the package: linear regression model (`lm`), generalized linear modeling (`glm`), and projection pursuit regression (`ppr`).

- **nnet** (**?**Ripley 2012a): This package is a recommended package for R. It implements a multi-layer perceptron with one hidden layer (`nnet`), and uses a general quasi-Newton optimization procedure (the BFGS algorithm) for learning.

- **COREelearn** (Robnik-Sikonja and Savicky 2013): This package contains several learning techniques for classification and regression. We use the regression tree method (`regTree`) of the package here.

- **e1071** (Meyer *et al.* 2012): From this package, we use the available support vector machine (SVM), `svm`, to perform regression tasks.

The parameters of the methods considered in the experiments are shown in Table 3. We use the same parameter specifications for the two different datasets.

Methods	G. Furnace (RMSE)	M.-Glass (RMSE)	Methods	G. Furnace (RMSE)	M.-Glass (RMSE)
`randomForest`	0.91	0.016	`"SBC"`	0.72	0.022
`mlp`	0.86	0.011	`"DENFIS"`	0.89	0.101
`lm`	0.72	0.094	`"FIR.DM"`	1.23	0.234
`glm`	0.72	0.094	`"FS.HGD"`	0.83	0.052
`ppr`	0.64	0.050	`"GFS.THRIFT"`	1.64	0.225
`nnet`	**0.58**	**0.002**	`"GFS.FR.MOGUL"`	1.08	0.084
`regTree`	1.41	0.062	`"WM"`	0.78	0.019
`svm`	0.72	0.033	`"HYFIS"`	0.87	0.087
			`"ANFIS"`	0.64	0.032

Table 4: The results obtained in the regression tasks.

Name	Attributes	Patterns	Classes
`iris`	4	150	3
`pima`	8	768	2
`wine`	13	178	3

Table 5: Datasets considered for classification tasks.

Experimental results

This section presents the results of the methods considered for comparison. To evaluate the results, we calculate the RMSE. The complete results are shown in Table 4. It can be seen that the best result for the gas furnace dataset are an RMSE of 0.58, obtained by the `nnet` method. For the Mackey-Glass series, the best method is `nnet`, with an RMSE of 0.002. Based on this benchmarking experiment, we can say that the methods that provide the best three results for the gas furnace dataset are `nnet`, `"ANFIS"`, and `ppr`. One of these methods is from the **frbs** package. In the case of the Mackey-Glass series, methods from other packages like `nnet`, `mlp`, and `randomForest` outperform the methods included in package **frbs**. Generally, the methods included in package **frbs** obtain reasonable, competitive results.

5.2. Classification tasks

In this section an illustrative empirical study of FRBS methods in classification is provided. We describe again the experiment design, which includes the datasets, the methods considered for comparison, their parameters, and finally the experimental results.

Datasets

In these experiments, we consider three datasets, namely, the `iris`, `pima`, and `wine` datasets. Some properties of the datasets can be seen in Table 5. To validate the experiments, we consider a 5-fold cross-validation, i.e., we randomly split the datasets into five folds, each containing 20% of the patterns of the dataset. Then, we use four partitions for training and one partition for testing. All of the data are available from the KEEL dataset repository (Alcalá-Fdez, Fernandez, Luengo, Derrac, García, Sánchez, and Herrera 2011).

Methods considered for comparison and their parameters

Again, we compare the classification methods in the **frbs** package with several different packages available on CRAN. The methods are chosen since they are well-known methods and represent different characteristics in the way they solve the particular tasks. The packages used for comparison are the following ones:

- **COREElearn** (Robnik-Sikonja and Savicky 2013): As mentioned before, this package contains several methods. In this case, we use the k-nearest-neighbors classifier method (`knn`).
- **C50** (Kuhn, Weston, Coulter, and Quinlan 2012): The package implements the `C5.0` algorithm presented by Quinlan (1993).
- **randomForest** (Breiman *et al.* 2012): `randomForest` can be used both for regression and for classification, so that we use it also here.
- **nnet** (Ripley 2012a): We use `nnet` as in the regression task.
- **RSNNS** (Bergmeir and Benítez 2012): As in the regression task, we use `mlp` for classification.
- **tree** (Ripley 2012b): The package implements the `tree` method.
- **kernlab** (Karatzoglou, Smola, Hornik, and Zeileis 2004): The package implements SVM methods. In this experiment, we use the `ksvm` function to perform classification tasks.
- **fugeR** (Bujard 2012): The package implements the `fugeR` method which is a genetic algorithm to construct an FRBS model. We consider this package for comparison because it is a package already available from CRAN that applies FRBSs.

The parameters of the methods used in the experiments are shown in Table 6. The same parameter specifications were used for all the datasets in classification.

Experimental results

Table 7 shows the results obtained from the three experiments using 5-fold cross-validation. By considering all datasets, in these experiments the best results are obtained by `"FRBCS.CHI"`, `tree`, and `ksvm` for `iris`, `pima`, and `wine`, respectively. So, we see that the methods available in the **frbs** package can be considered competitive for classification tasks.

6. Other FRBS packages available on CRAN

In this section, we review in more detail the packages available on CRAN which implement FRBSs. We compare them to package **frbs**, considering functionality and capability. The following packages provide functions which are able to construct FRBSs.

Methods	Parameters
knn	none
C5.0	trial = 100
randomForest	importance = TRUE, proximity = TRUE
nnet	size = 5, rang = 0.8, decay = 5e-4, maxit = 1000
mlp	maxit = 350, learnFuncParams = $[0,1]$, size = 5
tree	none
ksvm	type = "C-bsvc", kernel = "rbfdot", kpar = list(sigma = 0.1), C = 10, prob.model = TRUE
fugeR	generation = 100, population = 100, elitism = 20, verbose = TRUE, threshold = NA, sensiW = 0.0, speciW = 0.0, accuW = 0.0, rmseW = 1.0, maxRules = 10, maxVarPerRule = 2, labelsmf = 3
"FRBCS.CHI"	num.labels = 9, type.mf = 3
"FRBCS.W"	num.labels = 9, type.mf = 3
"GFS.GCCL"	popu.size = 70, num.labels = 3, persen_cross = 0.9, max.gen = 100, persen_mutant = 0.3
"FH.GBML"	popu.size = 50, max.num.rule = 100, persen_cross = 0.9, max.gen = 100, persen_mutant = 0.3, p.dcare = 0.8, p.michigan = 1
"SLAVE"	num.labels = 5, persen_cross = 0.9, max.iter = 100, max.gen = 100, persen_mutant = 0.3, k.low = 0, k.upper = 1, epsilon = 0.7

Table 6: The parameters of the methods selected for comparison for classification.

Methods	Classification rate (%)		
	iris	pima	wine
knn	94.67	74.09	96.62
C5.0	94.00	74.34	94.35
randomForest	95.33	76.56	96.61
nnet	95.33	65.50	93.19
mlp	94.00	73.43	97.18
tree	94.67	**76.57**	92.67
ksvm	96.00	76.56	**98.29**
fugeR	95.33	76.09	89.31
"FRBCS.CHI"	**97.34**	67.44	92.67
"FRBCS.W"	96.00	69.92	92.67
"GFS.GCCL"	94.00	66.54	84.91
"FH.GBML"	95.34	68.62	81.93
"SLAVE"	97.33	72.91	88.17

Table 7: The results obtained in classification.

Package sets. As already stated briefly in Section 1, package **sets** (Meyer and Hornik 2009) provides standard procedures for the construction of sets, fuzzy sets, and multisets. Especially w.r.t. fuzzy sets, an advantage of package **sets** is that it does not only rely on the

R built-in `match()` function to perform set operations, but it also provides comprehensive operations such as negation, conjunction, disjunction, implication, etc. For example, the conjunction operator, `.T.()`, provides many options such as: `"Zadeh"`, `"drastic"`, `"product"`, `"Lukasiewicz"`, `"Fodor"`, `"Hamacher"`, `"Yager"`, etc. Furthermore, there are several functions to set the shape of the membership function which are `fuzzy_normal()` for the Gaussian function, `fuzzy_trapezoid()` for trapezoid, `fuzzy_triangular()` for a triangle shape, etc. Regarding the construction of FRBSs, package **sets** has the capability to perform fuzzy reasoning by using `fuzzy_inference()`, and to convert fuzzy into crisp values by using `gset_defuzzify()`. However, the package does not include learning algorithms, which is the main focus of our package. So, at first sight package **sets** may seem an ideal base for the implementation of the functionality available in our package. But there is only the Mamdani model available, and we found it difficult to extend the **sets** package to our needs, as the underlying data types and syntactics do not facilitate automatization of the construction process of FRBSs[2]. So, finally we opted for simple numerical matrices as the basic data type in the **frbs** package. In package **frbs**, we provide many different learning procedures to learn from numerical data, as well as a mechanism for fuzzy reasoning without learning, by using our function `frbs.gen()`. Furthermore, package **frbs** does not only implement the Mamdani model but it also has the TSK and FRBCS models implemented.

Package fugeR. The package **fugeR** (Bujard 2012) implements genetic algorithms to construct an FRBS from numerical data for classification. It is based on fuzzy cooperative coevolution (Peña Reyes 2002) where two co-evolving species are defined: the database and the rulebase. In this package, there are two main functions which are `fugeR.run()` for construction of the FRBS model and `fugeR.predict()` for prediction. So, package **fugeR** implements one particular classification method based on genetic algorithms. Our package implements the same workflow, but with more than ten different models, both for classification and regression, among them various different ones which use genetic algorithms.

7. Conclusions

This paper has presented the R package **frbs** for FRBSs. It implements the most commonly used types of FRBSs, namely, Mamdani and TSK kinds and several variants, for both classification and regression tasks. In addition it offers more than ten different learning methods to build FRBSs out of data in a homogeneous interface. An overview of the theory and implementation of all the methods, as well as an example of the usage of the package, and comparisons to other packages available on CRAN is offered. The comparison was made both experimentally, by considering other packages that use different approaches for regression and classification, as well as functionally, considering packages that implement FRBSs. The aim of the package is to make the most widely used learning methods for FRBSs available to the R community, in particular, and to the computational intelligence community of users and practitioners, in general.

[2] Actually we tried pretty hard but did not find a way to get the parameters to `fuzzy_inference()` evaluated, as they are passed to `substitute` internally by that function.

Acknowledgments

This work was supported in part by the Spanish Ministry of Science and Innovation (MICINN) under Projects TIN2009-14575, TIN2011-28488, TIN2013-47210-P, and P10-TIC-06858. Lala S. Riza would like to express his gratitude to the Dept. of Computer Science, Universitas Pendidikan Indonesia, for supporting him to pursue the Ph.D. program, and to the Directorate General of Higher Education of Indonesia, for providing a Ph.D. scholarship. The work was performed while C. Bergmeir held a scholarship from the Spanish Ministry of Education (MEC) of the "Programa de Formación del Profesorado Universitario (FPU)".

References

Alcalá-Fdez J, Fernandez A, Luengo J, Derrac J, García S, Sánchez L, Herrera F (2011). "KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework." *Journal of Multiple-Valued Logic and Soft Computing*, **17**(2–3), 255–287.

Babuska R (1998). *Fuzzy Modeling for Control*. Kluwer Academic Press.

Bai Y, Zhuang H, Roth ZS (2005). "Fuzzy Logic Control to Suppress Noises and Coupling Effects in a Laser Tracking System." *IEEE Transactions on Control Systems Technology*, **13**(1), 113–121.

Bergmeir C, Benítez JM (2012). "Neural Networks in R Using the Stuttgart Neural Network Simulator: **RSNNS**." *Journal of Statistical Software*, **46**(7), 1–26.

Box G, Jenkins GM (1970). *Time Series Analysis: Forecasting and Control*. CA: Holden Day.

Boyacioglu MA, Avci D (2010). "An Adaptive Network-Based Fuzzy Inference System (ANFIS) for the Prediction of Stock Market Return: The Case of the Istanbul Stock Exchange." *Expert Systems with Applications*, **37**(12), 7908–7912.

Breiman L, Cutler A, Liaw A, Wiener M (2012). ***randomForest***: *Breiman and Cutler's Random Forest for Classification and Regression*. R package version 4.6-7, URL http://www.stat-www.berkeley.edu/users/breiman/RandomForests.

Buckley JJ, Hayashi Y (1994). "Fuzzy Neural Networks: A Survey." *Fuzzy Sets and Systems*, **66**, 1–13.

Bujard A (2012). ***fugeR***: *Fuzzy Genetic, a Machine Learning Algorithm to Construct Prediction Model Based on Fuzzy Logic*. R package version 0.1.2, URL http://CRAN.R-project.org/package=fugeR.

Chi Z, Yan H, Pham T (1996). *Fuzzy Algorithms with Applications to Image Processing and Pattern Recognition*. World Scientific. ISBN 9810226977.

Chiu S (1996). "Method and Software for Extracting Fuzzy Classification Rules by Subtractive Clustering." *Fuzzy Information Processing Society, NAFIPS*, pp. 461–465.

Cordon O, Herrera F, Hoffmann F, Magdalena L (2001). *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*. Singapore: World Scientific Publishing.

Gonzalez A, Peréz R (2001). "Selection of Relevant Features in a Fuzzy Genetic Learning Algorithm." *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, **31**(3), 417–425.

González A, Pérez R, Verdegay JL (1993). "Learning the Structure of a Fuzzy Rule: A Genetic Approach." In *Proc. First European Congress on Fuzzy and Intelligent Technologies (EUFIT'93)*, pp. 814–819.

Herrera F, Lozano M, Verdegay J (1998). "A Learning Process for Fuzzy Control Rules Using Genetic Algorithms." *Fuzzy Sets and Systems*, **100**, 143–158.

Ichihashi H, Watanabe T (1990). "Learning Control System by a Simplified Fuzzy Reasoning Model." In *IPMU '90*, pp. 417–419.

Ishibuchi H, Nakashima T (2001). "Effect of Rule Weights in Fuzzy Rule-Based Classification Systems." *IEEE Transactions on Fuzzy Systems*, **1**, 59–64.

Ishibuchi H, Nakashima T, Murata T (1999). "Performance Evaluation of Fuzzy Classifier Systems for Multidimensional Pattern Classification Problems." *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, **29**(5), 601–618.

Ishibuchi H, Nakashima T, Nii M (2005a). *Classification and Modeling with Linguistic Information Granules: Advanced Approaches to Linguistic Data Mining*. Springer-Verlag. ISBN 3-540-20767-8.

Ishibuchi H, Nozaki K, Tanaka H (1992). "Distributed Representation of Fuzzy Rules and Its Application to Pattern Classification." *Fuzzy Sets and Systems*, **52**, 21–32.

Ishibuchi H, Nozaki K, Tanaka H (1994). "Empirical Study on Learning in Fuzzy Systems by Rice Taste Analysis." *Fuzzy Sets and Systems*, **64**(2), 129–144.

Ishibuchi H, Yamamoto T, Nakashima T (2005b). "Hybridization of Fuzzy GBML Approaches for Pattern Classification Problems." *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, **35**(2), 359–365.

Jang JSR (1993). "ANFIS: Adaptive-Network-Based Fuzzy Inference System." *IEEE Transactions on Systems, Man, and Cybernetics*, **23**(3), 665–685.

Karatzoglou A, Smola A, Hornik K, Zeileis A (2004). "**kernlab** – An S4 Package for Kernel Methods in R." *Journal of Statistical Software*, **11**(9), 1–20.

Kasabov NK, Song Q (2002). "DENFIS: Dynamic Evolving Neural-Fuzzy Inference System and Its Application for Time-Series Prediction." *IEEE Transactions on Fuzzy Systems*, **10**(2), 144–154.

Kim J, Kasabov N (1999). "HyFIS: Adaptive Neuro-Fuzzy Inference Systems and Their Application to Nonlinear Dynamical Systems." *Neural Networks*, **12**(9), 1301–1319.

Klir GJ, Yuan B (1995). *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice Hall PTR. ISBN 0-13-101171-5.

Kosko B (1992). "Fuzzy Systems as Universal Approximators." In *FUZZ-IEEE'92*, pp. 1153–1162.

Kuhn M, Weston S, Coulter N, Quinlan R (2012). ***C50**: C5.0 Decision Trees and Rule-Based Models*. R package version 0.1.0-013, URL http://cran.r-project.org/web/packages/C50.

Lewin A (2007). ***fuzzyFDR**: Exact Calculation of Fuzzy Decision Rules for Multiple Testing*. R package version 1.0, URL http://CRAN.R-project.org/package=fuzzyFDR.

Mackey MC, Glass L (1977). "Oscillation and Chaos in Physiological Control Systems." *Science*, **197**(4300), 287–289.

Mamdani EH (1974). "Applications of Fuzzy Algorithm for Control a Simple Dynamic Plant." *Proceedings of the Institution of Electrical Engineers*, **121**(12), 1585–1588.

Mamdani EH, Assilian S (1975). "An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller." *Int. J. Man Mach. Stud*, **7**, 1–13.

Mandal DP, Murthy CA, Pal SK (1992). "Formulation of a Multivalued Recognition System." *IEEE Transactions on Systems, Man, and Cybernetics*, **22**, 607–620.

Meyer D, Dimitriadou E, Hornik K, Weingessel A, Leisch F (2012). ***e1071**: Misc Functions of the Department of Statistics (e1071), TU Wien*. R package version 1.6-1, URL http://CRAN.R-project.org/package=e1071.

Meyer D, Hornik K (2009). "Generalized and Customizable Sets in R." *Journal of Statistical Software*, **31**(2), 1–27.

Nomura H, Hayashi l, Wakami N (1992). "A Learning Method of Fuzzy Inference Rules by Descent Method." *IEEE International Conference on Fuzzy Systems*, pp. 203–210.

Peña Reyes CA (2002). *Coevolutionary Fuzzy Modelling*. Master's thesis, Faculté Informatique et Communications, École Polytechnique Fédérale De Lausanne. URL http://library.epfl.ch/en/theses/?nr=2634.

Pedrycz W (1996). *Fuzzy Modelling: Paradigms and Practice*. Kluwer Academic Press.

Pedrycz W, Gomide F (1998). *An Introduction to Fuzzy Sets: Analysis and Design*. The MIT Press.

Quinlan R (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers. URL http://www.rulequest.com/see5-unix.html.

Ripley B (2012a). ***nnet**: Feed-Forward Neural Networks and Multinomial Log-Linear Models*. R package version 7.3-5, URL http://www.stats.ox.ac.uk/pub/MASS4.

Ripley B (2012b). ***tree**: Classification and Regression Trees*. R package version 1.0-33, URL http://cran.r-project.org/web/packages/tree.

Riza LS, Bergmeir C, Herrera F, Benítez JM (2014). ***frbs**: Fuzzy Rule-Based Systems for Classification and Regression Tasks*. R package version 2.2-0, URL http://CRAN.R-project.org/package=frbs.

Riza LS, Bergmeir C, Herrera F, Benítez JM (2015). "frbs: Fuzzy Rule-Based Systems for Classification and Regression in R." *Journal of Statistical Software*, **65**(6), 1–30. URL http://www.jstatsoft.org/v65/i06/.

Robnik-Sikonja M, Savicky P (2013). ***CORElearn**: CORElearn Classification, Regression, Feature Evaluation and Ordinal Evaluation*. R package version 0.9.41, URL http://lkm.fri.uni-lj.si/rmarko/software/.

Sugeno M, Kang GT (1988). "Structure Identification of Fuzzy Model." *Fuzzy Sets and Systems*, **28**, 15–33.

Sugeno M, Yasukawa T (1993). "A Fuzzy-Logic-Based Approach to Qualitative Modeling." *IEEE Transactions on Fuzzy Systems*, **1**(1), 7–31.

Takagi T, Sugeno M (1985). "Fuzzy Identification of Systems and Its Applications to Modeling and Control." *IEEE Transactions on Systems, Man, and Cybernetics*, **51**(1), 116–132.

Thrift P (1991). "Fuzzy Logic Synthesis with Genetic Algorithms." In *Proc. of the Fourth International Conf. on Genetic Algorhtms (ICGA91)*, pp. 509–513.

Wang LX (1992). "Fuzzy Systems Are Universal Approximators." In *FUZZ-IEEE'92*, pp. 1163–1170.

Wang LX (1994). *Adaptive Fuzzy Systems and Control: Design and Analysis*. Prentice-Hall.

Wang LX, Mendel JM (1992). "Generating Fuzzy Rules by Learning from Examples." *IEEE Transactions on Systems, Man, and Cybernetics*, **22**(6), 1414–1427.

Wuerts D, et al (2012). ***fRegression**: Regression Based Decision and Prediction*. R package version 2160.77, URL http://www.rmetrics.org.

Yager R, Filev D (1994). "Generation of Fuzzy Rules by Mountain Clustering." *Journal of Intelligent and Fuzzy Systems*, **2**(3), 209–219.

Zadeh L (1965). "Fuzzy Sets." *Information and Control*, **8**, 338–353.

Zadeh L (1975). "The Concept of a Linguistic Variable and its Application to Approximate Reasoning - Part I." *Information Sciences*, **8**(3), 199–249.

Zhou SM, Lyons RA, Brophy S, Gravenor MB (2012). "Constructing Compact Takagi-Sugeno Rule Systems: Identification of Complex Interactions in Epidemiological Data." *PLoS ONE*, **1**(12), 1–14.

Affiliation:

Lala Septem Riza, Christoph Bergmeir, Francisco Herrera, José Manuel Benítez
Department of Computer Science and Artificial Intelligence
E.T.S. de Ingenierías Informática y de Telecomunicación
CITIC-UGR, IMUDS, University of Granada
18071 Granada, Spain

E-mail: lala.s.riza@decsai.ugr.es, c.bergmeir@decsai.ugr.es, herrera@decsai.ugr.es, j.m.benitez@decsai.ugr.es
URL: http://dicits.ugr.es/, http://sci2s.ugr.es/