Information Sciences 287 (2014) 68–89

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Implementing algorithms of rough set theory and fuzzy rough set theory in the R package "RoughSets"

Lala Septem Riza [a,*], Andrzej Janusz [b], Christoph Bergmeir [a], Chris Cornelis [a], Francisco Herrera [a], Dominik Ślęzak [b], José Manuel Benítez [a]

[a] *Department of Computer Science and Artificial Intelligence, CITIC-UGR, IMUDS, University of Granada, Spain*
[b] *Institute of Mathematics, University of Warsaw, Poland*

A R T I C L E   I N F O

Article history:
Received 24 February 2014
Received in revised form 12 June 2014
Accepted 24 July 2014
Available online 1 August 2014

Keywords:
Rough set
Fuzzy rough set
Instance selection
Discretization
Feature selection
Rule induction

A B S T R A C T

The package *RoughSets*, written mainly in the R language, provides implementations of methods from the rough set theory (RST) and fuzzy rough set theory (FRST) for data modeling and analysis. It considers not only fundamental concepts (e.g., indiscernibility relations, lower/upper approximations, etc.), but also their applications in many tasks: discretization, feature selection, instance selection, rule induction, and nearest neighbor-based classifiers. The package architecture and examples are presented in order to introduce it to researchers and practitioners. Researchers can build new models by defining custom functions as parameters, and practitioners are able to perform analysis and prediction of their data using available algorithms. Additionally, we provide a review and comparison of well-known software packages. Overall, our package should be considered as an alternative software library for analyzing data based on RST and FRST.

© 2014 Elsevier Inc. All rights reserved.

1. Introduction

Rough set theory (RST) was introduced by Pawlak in 1982 [75] as a methodology for data analysis based on the approximation of concepts in information systems. It revolves around the notion of discernibility: the ability to distinguish between objects, based on their attribute values. Given an indiscernibility relation, we can construct lower and upper approximations of concepts. Objects included in the lower approximation can be classified with certainty as members of the concept. In contrast, the upper approximation contains objects possibly belonging to the concept. For more than three decades RST has been attracting researchers and practitioners in many different areas. For example, RST is applied in diverse domains such as water quality analysis [52], intrusion detection [63], bioinformatics [57], and character pattern recognition [60]. An important advantage of RST is that it does not require additional parameters to analyze the data.

RST has been generalized in many ways to tackle various problems. In particular, in 1990, Dubois and Prade [19] combined concepts of vagueness expressed by membership degrees in fuzzy sets [117] and indiscernibility in RST to obtain fuzzy rough set theory (FRST). FRST allows partial membership of an object to the lower and upper approximations, and moreover, approximate equality between objects can be modeled by means of fuzzy indiscernibility relations. An advantage of this is that we do not need to perform discretization if our data contain real-valued attributes. FRST has been used e.g., for feature selection, instance selection, classification, and regression. There are many application areas that have been addressed by FRST, see e.g., [17,38,116].

* Corresponding author.
E-mail addresses: lala.s.riza@decsai.ugr.es (L.S. Riza), J.M.Benitez@decsai.ugr.es (J.M. Benítez).

http://dx.doi.org/10.1016/j.ins.2014.07.029
0020-0255/© 2014 Elsevier Inc. All rights reserved.

Software packages are essential for an effective deployment of these techniques as well as to facilitate further research. To date, several software packages for both theories are already available. Rough Set Data Explorer (ROSE) is an RST-based software system created by the Laboratory of Intelligent Decision Support Systems of the Institute of Computing Science in Poznań [81,82]. In [3,4], a free software system for data exploration, classification support, and knowledge discovery called the Rough Set Exploration System (RSES) was presented. The rough set toolkit for analysis of data (ROSETTA), which is an advanced system for RST data analysis [72,73], includes the RSES library as the computation kernel. Also, a few algorithms from FRST have been implemented in the Waikato Environment for Knowledge Analysis (WEKA) [42]. WEKA is a collection of machine learning algorithms for data mining tasks implemented in Java [33]. Rough Set Based Intelligent Data Analysis System (RIDAS) is another data mining toolkit utilizing notions from RST [109]. It was developed at Chongqing University of Posts and Telecommunications and consists of a C++ kernel and a GUI for Windows systems. An important data mining system for inducing decision rules from various types of data, called Learning from Examples based on Rough Sets (LERS), was created at University of Kansas [30]. There have also been developed a few rule based expert systems for a medical diagnostics purposes. One of the most prominent is PRIMEROSE [106] which allows generation of decision and inhibitory rules to construct reliable differential diagnosis.

Generally, the available software packages only consider the implementations of discretization, feature selection, and rule induction algorithms which can be used by practitioners to address their problems. However, there are no packages that provide comprehensive facilities for an implementation of fundamental concepts for academic purposes and further research. Therefore, this paper presents the *RoughSets* package that allows researchers and practitioners to explore both the basic knowledge of the theories and their applications. It was written mainly in the R language [39,104].

R is a widely used analysis environment for scientific computing and visualization, such as statistics, data mining, bioinformatics, and machine learning. Currently, over 5000 packages are included in the repositories of the Comprehensive R Archive Network (CRAN) at http://cran.r-project.org/ and the Bioconductor project at http://www.bioconductor.org/. Every package submitted into the repositories is checked to meet some quality standards, such as representative documentation and running on any operating systems (e.g., MS Windows, Mac OS X, Linux). Furthermore, according to a popularity survey conducted by Muenchen [66], R has been the most popular tool used for data analytics, data mining, and big data in 2013. In this setting, it seems quite natural to design and develop a complete and solid package for RST and FRST in R, which is the main motivation for the software that we present in this paper. The *RoughSets* package is available on CRAN at http://cran.r-project.org/package=RoughSets.

The remainder of this paper is structured as follows. Section 2 gives a brief overview of RST and FRST. Section 3 presents the main applications of both theories. In Section 4, we discuss the package architecture and capabilities of the package in detail. Section 5 shows examples of the usage of the package. We provide a review and comparison with currently available packages based on their capabilities and functionalities in Section 6. Finally, Section 7 concludes the paper.

2. Rough set and fuzzy rough set preliminaries

In this section, we review some basic notions related to RST and FRST. In particular, we focus on the indiscernibility relation, the lower and upper approximations, the positive and boundary region, and the decision-relative discernibility matrix. For further details on basic concepts and extensions of RST, interested readers are referred to [77–79], whereas some good introductions to FRST include e.g., [13,19,20,114,115].

We first introduce some notations which are used throughout the paper. A dataset is represented in terms of an information system $\mathcal{A} = (U, A)$ [74], where U is a finite, non-empty set of objects called the universe of discourse[1] and A is a finite, non-empty set of attributes, such that $a : U \to V_a$ for every $a \in A$, where V_a is the set of values that the attribute a may take. A decision system is a special kind of information system, used in the context of classification and prediction, in which d is a designated attribute called the decision attribute, and the attributes in A are called conditional attributes. More formally, it is a pair $\mathcal{A} = (U, A \cup \{d\})$, where $d \notin A$ is the decision attribute. For RST and some of the FRST methods, the decision d has to be nominal. In the other FRST methods, the decision can be nominal or real-valued. The methods in the *RoughSets* package are implemented accordingly, allowing real-valued decisions where it is possible.

2.1. Indiscernibility relation

The main notion of RST is the indiscernibility relation.[2] Basically, it can be understood as a relation showing to what extent two objects are identical or similar. In the following subsections, we explain how to construct such relations in RST and FRST.

2.1.1. RST

Pawlak [75] considered an equivalence relation to model indiscernibility. Given an information system (U, A), for any $B \subseteq A$ the equivalence relation R_B is defined by

[1] In table format, it refers to all instances, experiments or rows.

[2] There exist also alternative interpretations of RST, where the notion of indiscernibility is replaced e.g. by the dominance relation (see e.g. [26]). These interpretations are not considered in this paper.

$$R_B(x,y) = \{(x,y) \in U^2 | \forall a \in B, a(x) = a(y)\}. \tag{1}$$

If $(x,y) \in R_B(x,y)$, then x and y have exactly the same values for the attributes in B. The equivalence classes of this so-called B-indiscernibility relation are denoted by $[x]_B$.

More generally, indiscernibility can be modeled by means of a tolerance relation, i.e., a reflexive and symmetric binary relation in U. Such relations have been considered e.g. in [71,80,96], but are not implemented in the current package.

2.1.2. FRST

In FRST, it is assumed that R is at least a fuzzy tolerance relation in U, that is, R satisfies.

- Reflexivity: $\forall x \in U, R(x,x) = 1$.
- Symmetry: $\forall x,y \in U, R(x,y) = R(y,x)$.

Sometimes, additionally the following condition is imposed:

- $\mathcal{T}$-transitivity: $\forall x,y,z \in U, \mathcal{T}(R(x,y), R(y,z)) \leqslant R(x,z)$.

In this case, R is called a fuzzy $\mathcal{T}$-equivalence relation or $\mathcal{T}$-similarity relation. In a case when $\mathcal{T} = \min$, R is simply called a fuzzy equivalence relation or similarity relation.

For example, Hu et al. [36] considered the following kernel-based fuzzy relations:

- Gaussian kernel: $R(x,y) = \exp\left(-\frac{\|x-y\|^2}{\delta}\right)$.
- Exponential kernel: $R(x,y) = \exp\left(-\frac{\|x-y\|}{\delta}\right)$.
- Rational quadratic kernel: $R(x,y) = 1 - \frac{\|x-y\|^2}{\|x-y\|^2+\delta}$.
- Circular kernel: if $\|x-y\| < \delta, R(x,y) = \frac{2}{\pi}\arccos\left(\frac{\|x-y\|}{\delta}\right) - \frac{2}{\pi}\frac{\|x-y\|}{\delta}\sqrt{1-\left(\frac{\|x-y\|}{\delta}\right)^2}$.
- Spherical kernel: if $\|x-y\| < \delta, R(x,y) = 1 - \frac{3}{2}\frac{\|x-y\|}{\delta} + \frac{1}{2}\left(\frac{\|x-y\|}{\delta}\right)^3$.

where $\delta > 0$. They showed that each of them is a $\mathcal{T}_{cos}$-similarity relation, where the t-norm $\mathcal{T}_{cos}$ is defined as, for a,b in $[0,1]$,

$$\mathcal{T}_{cos}(a,b) = max\left(ab - \sqrt{1-a^2}\sqrt{1-b^2}, 0\right),$$

Given a fuzzy tolerance relation R, it is always possible to transform it into a $\mathcal{T}$-similarity relation [67]. A simple procedure to calculate the min-transitive closure of R is shown in Algorithm 1.

Algorithm 1. Computing the min-transitive closure of a fuzzy relation R.

```
input  : A fuzzy relation $R$.
output: A min-transitive fuzzy relation $R^m$.
while $R^m \neq R$ do
    foreach $x, y \in U$ do
        $R^m(x,y) = \max(R(x,y), \max_{z \in U}(\min(R(x,z), R(z,y))))$
    end
    $R \leftarrow R^m$
end
```

A common way to construct a fuzzy B-indiscernibility relation for $B \subseteq A$ proceeds by considering a fuzzy tolerance relation R_a for each quantitative attribute a, such as the following equations considered by Jensen and Shen in [48]:

$$\begin{aligned} R_a(x,y) &= 1 - \frac{|a(x)-a(y)|}{|a_{max}-a_{min}|}, \\ R_a(x,y) &= \exp\left(-\frac{(a(x)-a(y))^2}{2\sigma_a^2}\right), \\ R_a(x,y) &= \max\left(\min(\frac{a(y)-a(x)+\sigma_a}{\sigma_a}, \frac{a(x)-a(y)+\sigma_a}{\sigma_a}), 0\right), \end{aligned} \tag{2}$$

where σ_a^2 is the variance of feature a. For a qualitative (i.e., nominal) attribute a, the classical manner of discerning objects is used, i.e., $R(x,y) = 1$ if $a(x) = a(y)$ and $R(x,y) = 0$, otherwise. Then we can define for any subset B of A, the B-indiscernibility relation by $R_B(x,y) = \mathcal{T}(\underbrace{R_a(x,y)}_{a \in B})$, where $\mathcal{T}$ is a t-norm.

2.2. Approximations

The indiscernibility relations described in the previous section are used in the definition of another basic concept of RST: lower and upper approximations.

First, we consider the crisp case where indiscernibility is modeled by an equivalence relation in U. Given $B \subseteq A, X \subseteq U$ can be approximated using the information in B by constructing B-lower and B-upper approximations of X:

$$R_B \downarrow X = \{x \in U | [x]_B \subseteq X\},$$
$$R_B \uparrow X = \{x \in U | [x]_B \cap X \neq \emptyset\}.$$

In FRST, following Radzikowska and Kerre [83], crisp lower and upper approximations are generalized by means of an implicator $\mathcal{I}$ and a t-norm $\mathcal{T}$. Given a fuzzy indiscernibility relation R_B and a fuzzy set X in U, we define

$$\begin{aligned}(R_B \downarrow X)(y) &= \inf_{x \in U} \mathcal{I}(R_B(x,y), X(x)),\\ (R_B \uparrow X)(y) &= \sup_{x \in U} \mathcal{T}(R_B(x,y), X(x)).\end{aligned} \tag{3}$$

Other approaches for defining lower and upper approximations in FRST have been proposed such as vaguely quantified rough sets (VQRS) [12], ordered weighted average rough sets (OWA) [15], fuzzy variable precision rough sets (FVPRS) [118], soft fuzzy rough sets (SFRS) [35], robust fuzzy rough sets (RFRS) [37], and β-precision fuzzy rough sets (β-PFRS) [86].

2.3. Regions and degree of dependency

In the context of a decision system $(U, A \cup \{d\})$, it is possible to consider the positive and boundary regions of $B \subseteq A$. They can be defined in terms of the B-lower and B-upper approximations of the equivalence classes $[x]_d$.

- The B-positive region: it is the (fuzzy) set of objects in U that can be certainly classified using the conditional attributes B:

$$POS_B = \bigcup_{x \in U} R_B \downarrow [x]_d.$$

- The B-boundary region: it is the (fuzzy) set of objects, $x \in U$, that can be possibly, but not certainly, classified using the conditional attributes in B:

$$BND_B = \bigcup_{x \in U} R_B \uparrow [x]_d \setminus \bigcup_{x \in U} R_B \downarrow [x]_d.$$

Furthermore, we can compute a degree of dependency of the decision attribute d on the set of conditional attributes B, by

$$\gamma_B = \frac{|POS_B|}{|U|}. \tag{4}$$

The decision table $\mathcal{A}$ is called *consistent* if $\gamma_A = 1$.

2.4. Discernibility matrix

Let $(U, \mathcal{A})$ be an information system. In RST, the discernibility matrix $M(\mathcal{A})$ is a symmetric $n \times n$ matrix whose elements (c_{ij}) are defined as:

$$c_{ij} = \{a \in A : a(x_i) \neq a(x_j)\} \quad \text{for } i,j = 1, \ldots, n.$$

In other words, c_{ij} contains those attributes for which objects x_i and x_j differ.

The discernibility matrix can be adapted to work with a decision system, and is then called decision-relative discernibility matrix. In RST, it is defined as follows:

$$c_{ij} = \begin{cases} \{a \in A : a(x_i) \neq a(x_j)\}, & \text{if } d(x_i) \neq d(x_j) \\ \emptyset, & \text{otherwise,} \end{cases} \tag{5}$$

While in RST the (decision-relative) discernibility matrix is uniquely defined, there exist various alternatives in FRST. For instance, Chen et al. [9] defined the decision-relative discernibility matrix in FRST as

$$c_{ij} = \begin{cases} \{a \in A : 1 - R_B(x_i, x_j) \leqslant \lambda_i\} & \text{where } \lambda_i = (R_B \downarrow [x_i]_d)(x_i), \quad \text{if } d(x_i) \neq d(x_j) \\ \emptyset, & \text{otherwise.} \end{cases} \tag{6}$$

While $M(\mathcal{A})$ based on RST is symmetric, in FRST it is not necessarily symmetric. Other approaches for constructing the decision-relative discernibility matrix based on FRST can be found in, e.g., [10,51,105,118].

3. Application areas of rough sets and fuzzy rough sets

RST and FRST have been used to deal with problems in many areas, sometimes in collaboration with other approaches. For example, Pawlak and Skowron [77] quote many applications that employ rough sets and boolean reasoning.

This section presents some application areas of both theories: discretization, feature selection, instance selection, rule induction, and nearest neighbor-based classifiers. We focus on them as they are the tasks that most frequently apply RST and FRST. Additionally, there are three important steps in data modeling and analysis: preprocessing, learning, and prediction.

3.1. Discretization

Discretization refers to an approach for converting real-valued attributes into nominal ones in information systems. It should be ensured that this approach maintains the discernibility between objects. Therefore, at the learning stage we may want to produce more general models which avoid overfitting at the prediction step.

Based on the perspective proposed in [18], approaches to the discretization can be classified with regard to three different criteria:

- *global* vs. *local*: it refers to whether approaches evaluate discretizing values over the whole continuous instance/attribute in the information system or localized regions of instances/attributes. For example, the study in [69] proposes both local and global approaches handling of continuous attributes in large data bases.
- *supervised* vs. *unsupervised*: it refers to whether approaches consider values of instances in the discretization process or not. A simple example of an unsupervised approach is an equal width interval method that works by dividing the range of continuous attributes into k equal intervals, where k is given.
- *static* vs. *dynamic*: it refers to whether approaches need a parameter for determining the number of labels/symbolic values or not. In other words, the dynamic approaches are generating the number automatically along the discretization process.

As a basic discretization method, we explain here in detail the maximal discernibility (MD) heuristics presented in [5]. The algorithm shown in Algorithm 2 requires $O(|A| \cdot |U|^3)$ for determining one cut value. Other methods can be found in, e.g., [5,69]. Let $\mathcal{A} = (U, A \cup \{d\})$ be a consistent decision table. An arbitrary attribute $a \in A$ defines a sequence $v_1^a < v_2^a < \cdots < v_{n_a}^a$, where $\{v_1^a, v_2^a, \ldots, v_{n_a}^a\} = \{a(x) : x \in U\}$ and $n_a \leqslant n$ with n the number of instances. Then the set of all possible cuts (called basic cuts) on a is denoted by

$$C_a = \left\{ \left(a, \frac{v_1^a + v_2^a}{2}\right), \left(a, \frac{v_2^a + v_3^a}{2}\right), \ldots, \left(a, \frac{v_{n_a-1}^a + v_{n_a}^a}{2}\right) \right\}.$$

The set of possible cuts on all attributes is denoted by

$$C_A = \cup_{a \in A} C_a. \tag{7}$$

Algorithm 2. The MD Heuristics.

```
input  : A decision table $\mathcal{A} = (U, A \cup \{d\})$.
output: A set of cuts $\mathcal{C}$.
$\mathcal{C} \leftarrow \{\}$;
Calculate the set of basic cuts $C_A$ based on Equation 7;
Construct a binary matrix $\mathcal{L}$ which is a relation between the set of pairs
$(x_i, x_j)$ of objects discerned by $d$ and the set of basic cuts $C_A$. Each element of
the matrix is defined by
$e(i,j) = \begin{cases} 1, & \text{if a pair of objects on the } i^{th}\text{row can be discerned by cut values on the } j^{th}\text{column,} \\ 0, & \text{otherwise;} \end{cases}$
while $\mathcal{L} \neq \oslash$ do
    Select the cut $c_{max} \in C_A$ by maximization of the column of $\mathcal{L}$;
    Add $c_{max}$ to $\mathcal{C}$ and delete the $c_{max}$ column of $\mathcal{L}$ together with all rows
    having 1 in it;
end
```

3.2. Feature selection

Feature selection is a process of finding a subset of attributes which represents the same information as the complete feature set. In other words, a purpose of the feature selection is to identify significant attributes and to eliminate the dispensable ones. An attribute $a \in B \subseteq A$ can be regarded as dispensable in B if $R_B = R_{B \setminus \{a\}}$ otherwise a is called indispensable in B.

Furthermore, in both RST and FRST the feature selection typically refers to finding a *reduct* or a *superreduct*. A superreduct is a set of attributes $B \subseteq A$, such that $R_B = R_A$, where R_B and R_A are the indiscernibility relations defined by B and A, respectively [76,79]. If it is also minimal (w.r.t. inclusion), then it is called a reduct. The intersection of all reducts is called the *core*.

In this section, we focus on calculating a reduct of a decision table which is called *decision reduct*. A decision reduct of $\mathcal{A} = (U, A \cup \{d\})$ is a minimal (w.r.t inclusion) non-empty set of attributes $B \subseteq A$ such that $\delta_B = \delta_A$, where δ_A is the mapping on U such that for any object x it specifies all rows in the table whose attribute values are the same as for x, and then collects the decision values from each row [79]. Therefore, we can transform the problem of feature selection into looking for criteria that measure the quality of a set of features. Based on the measurement criteria in [93], we can evaluate sets of features to be decision reducts by the following approaches:

1. Conditional independence: Following [94], we define B as a decision reduct iff for any $u \in U$ we have following equality of probabilities P:

$$P_{\mathcal{A}}(d(u)/B(u)) = P_{\mathcal{A}}(d(u)/A(u)),$$

where

$$P_{\mathcal{A}}(d(u)/A(u)) = \frac{|\{u \in U : A(u) = w \wedge d(u) = w'\}|}{|\{u \in U : A(u) = w\}|},$$

for $w \in V_A^U$ and $w' \in V_d^U$.

2. Degrees of consistency: By considering the degree of dependency in Eq. (4), a set of attributes $B \in A$ is a decision reduct iff

$$\gamma_B = \gamma_A.$$

3. Approximate entropy: By considering *entropy* as a measure of information [90], we say that $B \in A$ is a decision reduct, iff

$$H_{B(u)}(d(u)) + log_2(1-\epsilon) \leqslant H_{A(u)}(d(u)),$$

where

$$H_{B(u)}(d(u)) = -\frac{1}{|U|}\sum_{u \in U} log_2 P_{B(u)}(d(u)),$$

and for $\epsilon \in [0,1)$.

4. Discernibility relation: We construct the decision-relative discernibility matrix which is based on Eqs. (5) and (6) for RST and FRST. In order to generate decision reducts, we calculate the discernibility function $f_{\mathcal{A}}$ of the matrix. It is a boolean function of m boolean variables $\bar{a}_1, \ldots, \bar{a}_m$ corresponding to the attributes $a_1, \ldots, a_m$ respectively, and defined by

$$f_{\mathcal{A}}(\bar{a}_1, \ldots, \bar{a}_m) = \wedge\{\vee \bar{c}_{ij} : 1 \leqslant j < i \leqslant n, c_{ij} \neq \emptyset\}, \quad (8)$$

where $\bar{c}_{ij} = \{\bar{a} : a \in c_{ij}\}$. The decision reducts of A are then the prime implicants of the function $f_{\mathcal{A}}$. Detailed explanations are given in [92].

Many algorithms have been proposed to find reducts in both RST and FRST settings. According to the output produced by these algorithms, we may divide them into three groups: those that produce a superreduct, a set of reducts, or a single reduct.

- Superreduct. An example is the algorithm based on RST proposed by Shen and Chouchoulas [91] called the QuickReduct algorithm. According to Algorithm 3, it can be seen that the computation time depends on the number of attributes instead of the number of objects, which has NP complexity. Other methods based on RST can be found in, e.g., [41,93,113].

Algorithm 3. QuickReduct.

```
input  : A decision table $\mathcal{A} = (U, A \cup \{d\})$.
output: A superreduct $SR$.
$SR \leftarrow \{\}$;
repeat
    $T \leftarrow SR$;
    foreach $x \in (A - SR)$ do
        if $\gamma_{SR \cup \{x\}} > \gamma_T$ then $T \leftarrow SR \cup \{x\}$;
        $SR \leftarrow T$;
    end
until $\gamma_{SR} == \gamma_A$;
```

Based on FRST, the fuzzy QuickReduct algorithm, which is a modified QuickReduct, was proposed in [48]. To obtain the degree of dependency γ, we can calculate the degree by using many variants of lower and upper approximations, e.g., implicator/t-norm approach [11], VQRS [12], OWA [15], FVPRS [118], SFRS [35], RFRS [37], β-PFRS [86]. Intuitively, other modifications can also be made by changing the stopping criterion (as, e.g., in [6]), or by randomizing the features considered for reducing the computation time.

- Set of reducts. Basically, a set of reducts is obtained by constructing the decision-relative discernibility matrix and then employing the discernibility function. We can construct the matrix by Eqs. (5) and (6), and then compute the discernibility function using Eq. (8).
  Some approaches have been proposed to reduce computation time and introduce new equations for constructing the decision-relative discernibility matrix. For instance, [51] proposes a technique employing a propositional satisfiability (SAT) framework to extract a set of reducts from the decision-relative discernibility matrix based on FRST. Other approaches can be found in [9,10,105,118].
- Reduct. There exist two simple ideas to obtain a single reduct which are by considering methods generating a set of reducts and a superreduct. First, through methods constructing the decision-relative discernibility matrix we obtain a set of reducts. After that we can choose a single reduct from the resulting reducts. The other method is by employing algorithms generating a superreduct. Here, we include some procedures to eliminate features. The elimination process is iterated until obtaining a reduct. The algorithm proposed in [40] is an example for obtaining a single reduct based on permutation procedures employing the elimination process.

3.3. Instance selection

The aim of instance selection is to remove or replace noisy, superfluous, or inconsistent instances from training datasets but retain consistent ones at the same time. From the RST perspective, it refers to evaluating each object included in the boundary region. In other words, according to the evaluation, we preserve objects in lower approximations, but we change objects included in the boundary region, for example deleting them or changing their class labels to be consistent values. For example, the EditRS algorithm is an algorithm based on RST aimed to perform instance selection [7]. There are two proposed types: Edit1RS and Edit2RS. The Edit1RS algorithm only takes into account objects included in lower approximations. In other words, we delete all objects in the boundary region. The second one, instead of deleting objects in boundary regions, their class labels are changed based on the generalized editing method. It attempts to obtain a consistent decision table after changing the class labels.

In FRST, there exist some methods performing instance selection. The fuzzy-rough instance selection (FRIS) method proposes three algorithms that employ a positive region as a measurement tool for selecting instances [44]. If the value of the positive region of objects is less than a threshold value then the objects can be removed. The complete FRIS-1 algorithm is defined as follows:

Algorithm 4. Fuzzy-rough instance selection version 1 (FRIS-1).

```
input  : A decision table $\mathcal{A} = (U, A \cup \{d\})$;
A granularity parameter $\alpha$;
A threshold value $\tau$.
output: A set of selected objects $Y$.
$Y \leftarrow U$;
foreach $x \in U$ do
    if $(POS_A^{\alpha,U}(x) < \tau)$ then $Y \leftarrow Y - x$;
end
```

The algorithm uses the following indiscernibility relation:

$$R_a^{\alpha}(x,y) = max\left(0, 1 - \alpha\frac{|a(x) - a(y)|}{l(a)}\right),$$

where α is a level of granularity and $l(a)$ is the range of the attribute a. In order to select objects the algorithm offers linear complexity with respect to the number of objects in the dataset, while computation of the positive region needs $O(|A| \cdot |U|^2)$.

The FRIS method is improved in [108] to obtain Fuzzy Rough Prototype Selection (FRPS), a method specifically designed to optimize the accuracy of the k-nearest neighbor (kNN) algorithm. First, instances are ordered according to a measure based on FRST that evaluates the lack of predictive ability of the instances, and a wrapper approach is then used to decide which instances to select.

3.4. Rule induction

Knowledge can be represented in many different ways. Production rules (also called rule-based classifiers) are arguably the most popular knowledge representation. The general structure of such a rule is *IF*...*THEN*.... The *IF* part refers to the *predecessor*, the *THEN* part to the *successor* [79]. One advantage of building rules is that naturally the model is easy to interpret and manipulate. In this section, we focus on rule induction through RST and FRST techniques.

In RST, a rule for the decision table $\mathcal{A}$ is called a decision rule denoted by *IF* φ *THEN* $d = v$, where $\varphi \in \mathcal{C}(A, V_a)$. $\mathcal{C}(A, V_a)$ is a set of pairs of conditional attributes A and their corresponding values V_a that are connected by the propositional $\wedge$ (conjunction), $\vee$ (disjunction), and $\neq g$ (negation). The decision rule is *true* in $\mathcal{A}$ if, and only if, $\|\varphi_{\mathcal{A}}\| \subseteq \|d = v_{\mathcal{A}}\|$ where in this case, $\|.\|$ is the set of objects *matching* the decision rule [79].

For generating rules from data, decision rules of classes in d must meet two properties: *completeness* and *consistency* [64]. Completeness means that for every instance $u \in U$ from class of the attribute d there exists a decision rule representing u while consistency means that there are no two different decision rules that describe the same instance in U. Additionally, in rule induction approaches, there exist three different types: *minimum*, *exhaustive*, and *satisfactory requirements* [99]. In the first case, we generate the smallest number of decision rules that are adequate for describing all given instances. On the other hand, the exhaustive approach refers to algorithms that induce all possible decision rules, whereas the last case contains decision rules that meet the pre-defined requirements.

Many algorithms to induce rules from data based on RST have been proposed. For example, the learning system LERS (Learning from Examples based on RST) introduced the learning from example module, version 2 (LEM2) [27]. It produces a minimal set of rules which is based on computing a single local covering for each concept from the decision table. An improvement of LEM2 is introduced in [28] and the modified learning from examples module, version 2 (MLEM2) algorithm [29]. MLEM2 treats numerical and symbolic attributes differently. For numerical attributes, it computes cut values as in discretization. After that, MLEM2 continues calling the LEM2 processes. Furthermore, an improvement of MLEM2 which is a local version of the method can be found in [32]. The Explore algorithm generating an exhaustive set of rules was presented in [65,100].

In case of FRST, an algorithm attempting to combine rule induction and feature selection at the same time is proposed in [49]. Basically it inserts some steps to generate rules in the fuzzy QuickReduct algorithm [46]. In [119], a set of decision rules is induced by constructing the decision-relative discernibility matrix based on the fuzzy variable precision rough sets. There exist other methods proposed to improve previous algorithms such as an extension of LEM2 for FRST called FRLEM2 [59].

3.5. Nearest neighbor-based classifiers

The nearest neighbor-based classifiers were introduced first by Fix and Hodges in [24]. Then, in 1967 they were improved and made famous by Cover and Hart [16]. In supervised learning, the k-nearest neighbor-based classifiers are defined as methods that predict new data/patterns based on the most similar or nearest k patterns in training data. This section attempts to illustrate enhancements of k-nearest neighbors based on FRST.

The fuzzy-rough ownership nearest neighbor algorithm was proposed by Sarkar [87] and then improved in [88,89]. To avoid determining k by trial and error, it uses all training data to construct the fuzzy-rough ownership function. It uses a squared weighted distance between a test pattern and all training data, and constrained fuzzy membership.

The study in [45] considered a summation of lower and upper approximations to predict the class of new instances. It uses lower and upper approximations based on Eq. (3) and on VQRS with fuzzy tolerance relations in Eq. (2). The procedure is outlined in Algorithm 5.

Algorithm 5. Fuzzy-rough nearest neighbor (FRNN).

```
input  : A decision table $\mathcal{A} = (U, A \cup \{d\})$ as training data; $y$ is new data.
output: $Class$ is a predicted class.
$N \leftarrow NN(y, K)$;  /* $NN$ is the $k$-nearest neighbor algorithm [16] */
$\tau \leftarrow 0$; $Class \leftarrow \emptyset$;
foreach $C \in d$ do
    if $((R\downarrow C)(y) + (R\uparrow C)(y))/2 \geq \tau$ then
        $Class \leftarrow C$;
        $\tau \leftarrow ((R\downarrow C)(y) + (R\uparrow C)(y))/2$;
    end
end
```

The complexity of this algorithm for classification of one object is $O(|U| + K \cdot |d|)$.

The fuzzy rough positive region (POSNN) [107] algorithm considers the positive region to predict the class of new data. Basically, the following steps are used to determine the class of an object y:

1. Determine the set NN of the k-nearest neighbors of y over the training data.
2. Select the class for which

$$\frac{\sum_{x \in NN} R(x,y)C(x)POS(x)}{\sum_{x \in NN} R(x,t)}$$

is maximal; where R, C, and POS are a fuzzy relation, a class membership function, and the positive region, respectively. The class membership function [53] is defined as

$$C(x) = \begin{cases} 0.51 + \frac{n_C}{K} \cdot 0.49, & \text{if } x \text{ is in class } C \\ \frac{n_C}{K} \cdot 0.49, & \text{otherwise}, \end{cases}$$

with n_C the number of instances having class C in NN.

4. Getting started with the *RoughSets* package

4.1. An introduction to the RoughSets package

As we mentioned before, *RoughSets* is an R package integrating implementations of methods from RST and FRST in a single software library. Currently, it is available from CRAN in version 1.1–0, at http://cran.r-project.org/package=RoughSets. It is licensed under the terms of the GPL so that it can be redistributed or modified under that term.

There are more than 40 functions included in the package. Their names are based on three parts separated by points: *prefix*, *suffix*, and *middle*.

- *prefix*: This part shows that corresponding functions perform a particular task as follows:
  - *BC*: basic concepts of a certain theory.
  - *D*: discretization.
  - *FS*: feature selection.
  - *IS*: instance selection.
  - *RI*: rule induction.
  - *C*: nearest neighbor-based classifiers.
  - *SF*: support functions.
  - *X*: auxiliary functions.
- *suffix*: Two names in this part are *RST* and *FRST*. The suffix *RST* refers to rough set theory while *FRST* shows that the function is applied to fuzzy rough set theory. Additionally, some functions that do not have these suffixes are used for both the theories.
- *middle*: The actual function name. It can consist of more than one word separated by points.

For example, *BC.positive.reg.RST* is a function based on rough set theory used to calculate the positive region, which is a part of the basic concepts. Other functions that have names not based on the above rules are **S3** functions which are built-in functions in the R framework. For instance, *summary* and *predict* are used to summarize objects and predict new data, respectively.

Additionally, the package has embedded some datasets used to do experimental studies. They are already in the format required by the package (see Section 4.2.1). For example, the hiring dataset is a dataset used originally in [55]. We also provide the housing [34], wine [25], and pima [1] datasets.

A key advantage of using *RoughSets* is that one can easily compare the performance of the models against an extensive set of models already available through other R packages. We can also combine them towards a collaborative, improved solution. Furthermore, because of the GPL, other researchers may collaborate freely by providing useful feedback, contributing their methods to the package, and even modifying current methods to tackle their specific problems.

Detailed technical and usage information can be found in the package manual, available on the CRAN web page, and on our groups' web page of the package, available at http://dicits.ugr.es/software/RoughSets/.

4.2. The package architecture and implementation details

In this section, we discuss the package in more detail. As we mentioned before, the *RoughSets* package implements algorithms for handling several tasks, as shown in Fig. 1, which depicts the six tasks that have been considered. In the basic concepts, we have implemented eight functions, e.g., *BC.IND.relation.RST* used to calculate the indiscernibility relations based on RST. We include discretization methods based on RST that calculate cut values using static & dynamic, local & global, and supervised & unsupervised methods. The other tasks included in this package are instance selection, feature selection, rule induction, and nearest neighbor-based classifiers. While algorithms used for feature selection and rule induction are based on RST and FRST, instance selection and nearest neighbor-based classifiers are based on FRST.

Table 1 illustrates a list of functions included in the package together with their references. Sometimes more than one algorithm has been implemented in one function. The different choices are available through suitable parameter values.

For instance, *BC.LU.approximation.FRST* allows us to implement different approaches for constructing lower and upper approximations by assigning the parameter *type.LU*.

In the next section, we provide more detail about the standard format of datasets and implementations of basic concepts, data preprocessing, and learning and prediction steps. They are essential parts before analyzing data based on RST and FRST using the *RoughSets* package.

4.2.1. Converting a dataset into the DecisionTable format

A common way in the R language to represent data is to use the *data.frame* type. However, the *RoughSets* package provides a standard format, which is *DecisionTable*. It contains a dataset and its properties can be used to express either a decision table or an information system. Therefore, methods recognize whether an object is an information system or decision table according to its properties. For a decision table, its properties involve three attributes: attribute descriptions, types of attributes, and an index of the decision attribute. However, we only have the first two of the properties when constructing an information system. In addition, the attribute descriptions are composed of two parts: names of attributes and their data range. Types of attributes show whether a particular attribute has nominal (symbolic) or real values. An index shows the position of the decision attribute in a given dataset.

Fig. 2 shows that there are two functions used for converting a dataset into the *DecisionTable* format: *SF.read.DecisionTable* and *SF.asDecisionTable*. *SF.read.DecisionTable* is used when we have a dataset saved in a file. In other words, the function does not only do the conversion but also imports the dataset from a file. On the other hand, *SF.asDecisionTable* is used to convert a dataset that we create in the R environment. After having our dataset in the *DecisionTable* format, we can conduct data analysis using the *RoughSets* package. An example showing the usage of some functions can be found in Section 5.

4.2.2. Implementation of the basic concepts

Four basic concepts have been implemented in the package: indiscernibility relations, lower and upper approximations, the positive region, and the decision-relative discernibility matrix. They are important to be implemented since they are used by other algorithms.

Based on RST, we can see in Table 1 the four functions implementing the basic concepts. Whereas *BC.IND.relation.RST* is used to implement the indiscernibility relation, *BC.LU.approximation.RST*, *BC.positive.reg.RST*, and *BC.discernibility.mat.RST* calculate the approximations, the positive region, and the discernibility matrix, respectively. In the current version, we have implemented the indiscernibility relation, the lower and upper approximations, and the positive region based on [75] whereas the discernibility matrix is based on [92].

Regarding FRST, we have considered many approaches to calculate lower and upper approximations that are already explained in Section 2. The approaches can be selected by assigning certain parameters. Furthermore, we also enable the users to create their own functions to construct the approximations. Therefore, this package was designed not only for practitioners but also for researchers who attempt to construct new models. The following is a list of parameters that can be set by using user-defined functions:

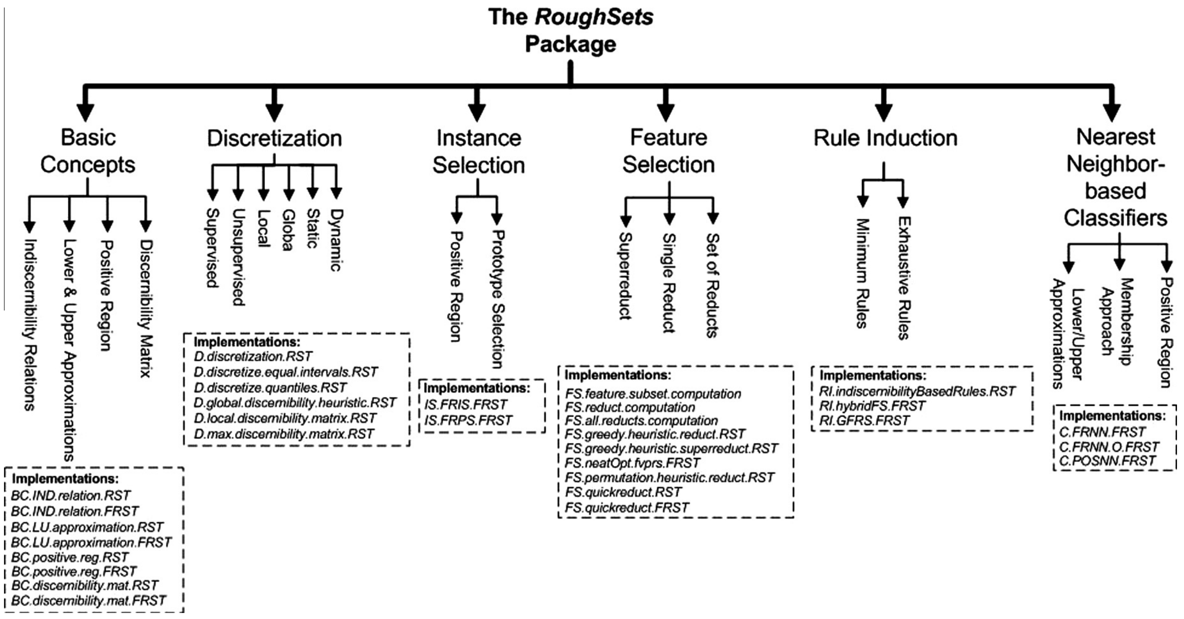

Fig. 1. Models and their implementation in *RoughSets*.

Table 1
Functions included in *RoughSets* and their references.

Names of functions	Descriptions	Refs.
BC.IND.relation.RST	Indiscernibility relation based on RST	[75]
BC.IND.relation.FRST	Indiscernibility relation based on FRST	[36,48,67]
BC.LU.approximation.RST	Approximations based on RST	[75]
BC.LU.approximation.FRST	Approximations based on FRST	[12,15,37,83,86,118]
BC.positive.reg.RST	Positive region based on RST	[75]
BC.positive.reg.FRST	Positive region based on FRST	[48]
BC.discernibility.mat.RST	Decision-relative discernibility matrix based on RST	[92]
BC.discernibility.mat.FRST	Decision-relative discernibility matrix based on FRST	[9,10,105,118]
D.max.discernibility.matrix.RST	Discretization based on maximal discernibility	[5]
D.local.discernibility.matrix.RST	Discretization based on local strategy	[5]
D.global.discernibility.heuristic.RST	Discretization based on global maximum discernibility heuristic	[69]
D.discretize.quantiles.RST	Discretization based on quantiles	[18]
D.discretize.equal.intervals.RST	Discretization based on equal interval size	[18]
FS.quickreduct.RST	Feature selection based on QuickReduct	[91]
FS.quickreduct.FRST	Feature selection based on fuzzy QuickReduct	[6,11,15,37,46,48,86,118]
FS.greedy.heuristic.superreduct.RST	Greedy heuristics for selecting a superreduct	[41,93,113]
FS.nearOpt.fvprs.FRST	Feature selection based on near-optimal reduction	[118]
FS.greedy.heuristic.reduct.RST	Greedy heuristics for selecting a reduct	[41,93,113]
FS.all.reducts.computation	Wrapper for computing all reducts	[9,10,92,105,118]
FS.permutation.heuristic.reduct.RST	Permutation heuristic for determining a reduct	[40]
IS.FRIS.FRST	Fuzzy rough instance selection	[44]
IS.FRPS.FRST	Fuzzy rough prototype selection	[108]
RI.indiscernibilityBasedRules.RST	Rule induction based on RST	[75]
RI.hybrid.FRST	QuickRules algorithm	[49]
RI.GFRS.FRST	Generalized fuzzy rough set rule induction	[119]
C.FRNN.FRST	Fuzzy-rough nearest neighbors	[45]
C.FRNN.O.FRST	Fuzzy-rough ownership nearest neighbors	[88]
C.POSNN.FRST	Positive region based fuzzy-rough nearest neighbors	[107]

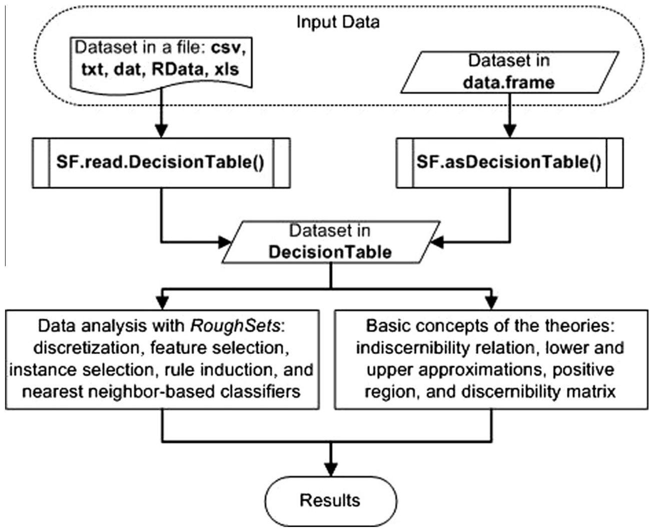
Input Data
Dataset in a file: csv, txt, dat, RData, xls
Dataset in data.frame
SF.read.DecisionTable()
SF.asDecisionTable()
Dataset in DecisionTable
Data analysis with RoughSets: discretization, feature selection, instance selection, rule induction, and nearest neighbor-based classifiers
Basic concepts of the theories: indiscernibility relation, lower and upper approximations, positive region, and discernibility matrix
Results

Fig. 2. Converting the dataset into the *DecisionTable* format.

- *type.relation*: it is a parameter representing a type of indiscernibility relation. We provide four types: *crisp*, *tolerance*, *transitive.kernel*, and *transitive.closure*. Detailed explanation about them can be read in the package manual. With the *type.relation* parameter set to *custom*, users can define their own functions.
- *type.aggregation*: it is a parameter for aggregation operators. There are three kinds: *crisp*, *t.tnorm*, and *custom*. Again, *custom* is used for a user-defined function as aggregating operator.
- *type.LU*: it is a parameter expressing a type of an approximation approach. It allows choosing one of the following types: *implicator.tnorm*, *vqrs*, *owa*, *fvprs*, *beta.pfrs*, *rfrs*, and *custom*. The *custom* value is chosen when we define our own functions for constructing lower and upper approximations.
- *t.tnorm*: it is a parameter for *t*-norm operators. There are six options in this parameter, such as *min*, *product*, etc. We can also create a user-defined function directly to replace the options.

- *t.implicator*: it is a parameter for implicator operators. As in *t.tnorm*, we can replace values of the parameter with a user-defined function.
- *w.owa*: this parameter allows to design a weight vector to construct approximations based on the OWA approach.

An example showing how to build a new model can be found in Section 5.

4.2.3. Implementation of preprocessing steps

As we explained above, the *RoughSets* package allows us to perform discretization, instance selection, and feature selection as preprocessing steps in data analysis. The functions involved in these tasks have been shown in Fig. 1 and Table 1.

Functions tackling preprocessing tasks in the package are not designed to generate a new *DecisionTable* automatically. In Fig. 3, it can be seen that while functions employed in discretization will produce cut values, functions working in feature selection and instance selection give decision reducts and indices of selected objects, respectively. Therefore, in order to generate a new *DecisionTable* we need to execute the function *SF.applyDecTable*. It produces a new *DecisionTable* considering the old decision table and outputs the preprocessing tasks as its parameters.

Furthermore, Fig. 4 shows an example of common preprocessing steps in data mining. It is obvious that we have many options to perform preprocessing. For example, we can consider discretization as a starting step and then continue to perform instance selection and feature selection in sequential order.

4.2.4. Implementation of learning and prediction steps

RoughSets considers two types of classifiers: rule-based and nearest neighbor-based classifiers. Their workflows can be seen in Fig. 5.

For rule-based classifiers, we need to execute rule induction functions for learning from data. In RST, we have the *RI.indiscernibilityBasedRules.RST* function while *RI.hybrid.FRST* and *RI.GFRS.FRST* are used for generating rules based on FRST. After calling the functions, we obtain a set of decision rules. We call this process the learning step. In order to predict new data, we need to execute the **S3** function *predict*.

Nearest neighbor-based classifiers do not separate learning and prediction steps. We only need to execute one function, e.g., *C.FRNN.FRST* to predict a new dataset based on the fuzzy-rough nearest neighbor algorithm. In other words, we supply both training and testing data in the functions as parameters. We provide the detailed explanation for how to use the functions in Section 5.

4.3. Installation of R and RoughSets

As of this writing, the current version of R is 3.0.2. To install the R platform, a user simply goes to the web site http://www.r-project.org and then follows the download instructions. After installing and running R, the ">" prompt will be shown in a command-line environment. The user can type commands or procedures interactively at the R prompt. Another way to use R is by creating scripts that contain commands and functions. A comprehensive introduction to the R language can be found in [103].

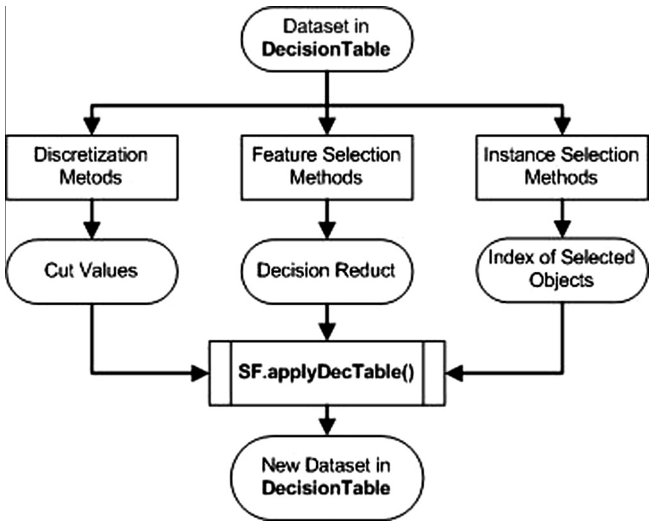
Dataset in DecisionTable
Discretization Metods
Feature Selection Methods
Instance Selection Methods
Cut Values
Decision Reduct
Index of Selected Objects
SF.applyDecTable()
New Dataset in DecisionTable

Fig. 3. Generating a new decision table in the *DecisionTable* format.

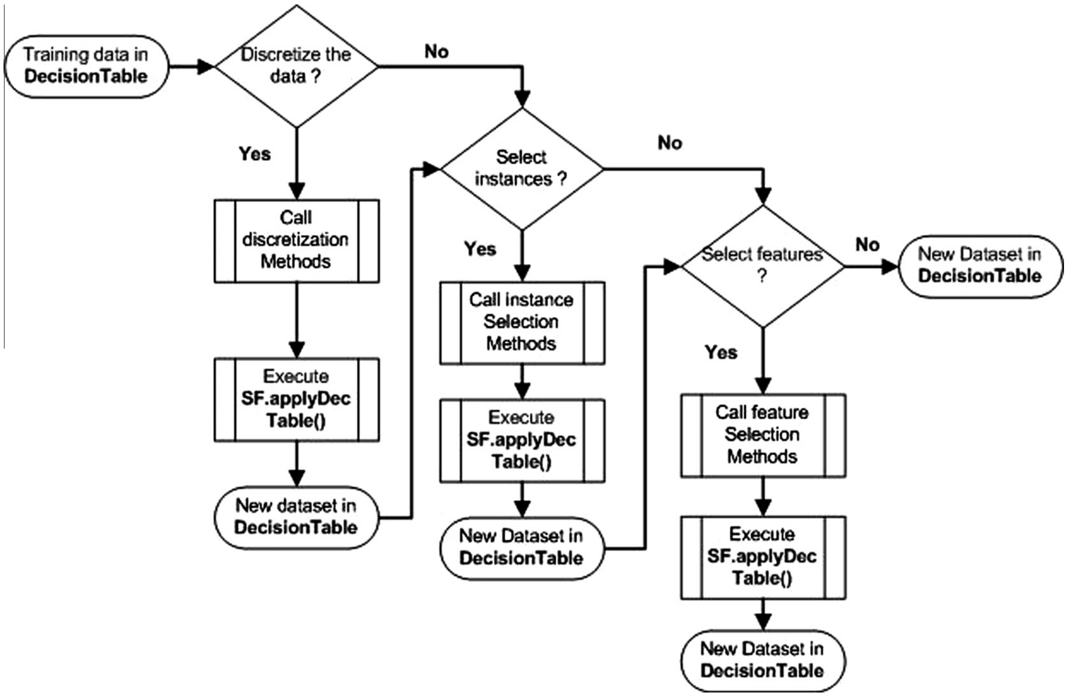
Training data in DecisionTable
Discretize the data ?
No
Yes
Call discretization Methods
Execute SF.applyDecTable()
New dataset in DecisionTable
Select instances ?
No
Yes
Call instance Selection Methods
Execute SF.applyDecTable()
New Dataset in DecisionTable
Select features ?
No
New Dataset in DecisionTable
Yes
Call feature Selection Methods
Execute SF.applyDecTable()
New Dataset in DecisionTable

Fig. 4. A scenario of data preprocessing using *RoughSets*.

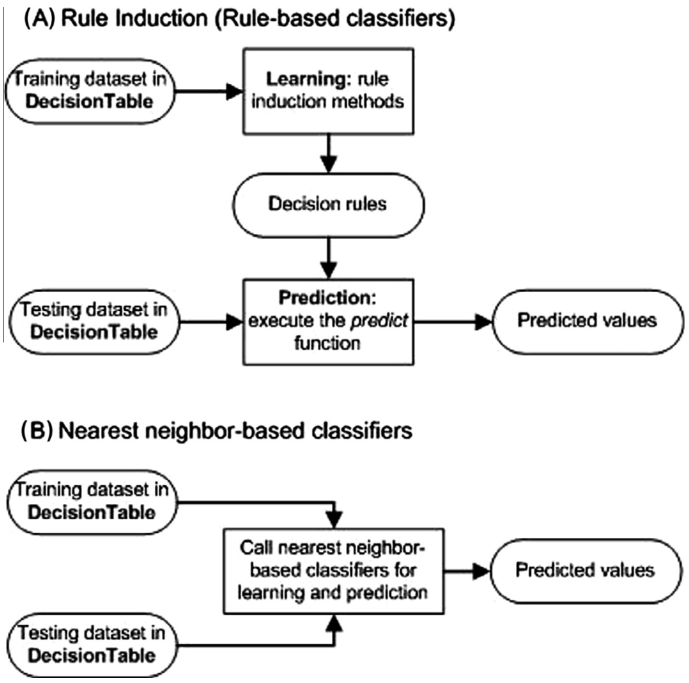
(A) Rule Induction (Rule-based classifiers)
Training dataset in DecisionTable
Learning: rule induction methods
Decision rules
Testing dataset in DecisionTable
Prediction: execute the predict function
Predicted values
(B) Nearest neighbor-based classifiers
Training dataset in DecisionTable
Call nearest neighbor-based classifiers for learning and prediction
Predicted values
Testing dataset in DecisionTable

Fig. 5. The learning and prediction schema in *RoughSets*: (a) rule induction (rule-based classifiers) and (b) nearest neighbor-based classifiers.

To install *RoughSets* from CRAN, users simply enter the following command in the R environment:

```
R > install.packages(c("sets", "class", "RoughSets"))
```

Here, *sets* and *class* are packages required for some functions of the *RoughSets* package.

The package installation has to be done only once. After that in any session using *RoughSets*, we need to load it with the following command:

```
R > library(RoughSets)
```

The command loads the *RoughSets* package and makes its functions available in the R environment. We can see a list of functions included in *RoughSets* by typing the following command:

```
R > library(help = RoughSets)
```

All R functions available in *RoughSets* are documented in the R help system in a hypertext format, and in the package manual in pdf format. The manual is available on the package website on CRAN. To get information on a particular function, we can call the help command, e.g.,

```
R > help(BC.IND.relation.RST)
```

5. Examples using *RoughSets*

In this section, we provide some examples showing how to use the *RoughSets* package. Simple data are used in the examples to give clear illustrations. It will be divided into three parts of examples: constructing the *DecisionTable* format, executing basic concepts, and applications.

5.1. Constructing datasets in the DecisionTable format

Every dataset used in the package has to be in *DecisionTable* format. Let *dt.ex1* be our dataset, the following code shows how to construct the *DecisionTable* of *dt.ex1*.

```
R > dt.exl <- data.frame(c(1,0,2,1,1,2,2,0), c(0.5,1.2,0.1,1.2,0.4,2.2,1.1,
 + 1.5), c(2,1,0,0,2,0,1,1), c(0,2,1,2,1,1,2,1))
R > colnames(dt.exl) <- c("a", "b", "c", "d")
R > DecTable. 1 <- SF.asDecisionTable(dataset = dt.exl, decision.attr = 4,
 + indx.nominal = c(1,3:4))
```

We can see *DecTable. 1* by typing:

```
R > print.default(DecTable 1)
```

To save space, we do not display the output of the previous command. Basically, *DecTable 1* (which is a *DecisionTable* object) contains the dataset as a *data.frame*, attribute descriptions (namely *desc.attrs*), type of attributes (namely *nominal.attrs*), and an index of the decision attribute (namely *decision.attr*). In *nominal.attrs*, we can see that the second attribute has real values and the others have nominal values. Furthermore, *desc.attrs* constitutes a set of attributes and their range of values, and *decision.attr* shows that the attribute on index 4 is the decision attribute. Detailed output for these examples and additional examples are available on our web page.

In case we have data with many rows and columns, it may be more convenient to construct *DecisionTable* from files, executing *SF.read.DecisionTable*. An example of this case can be found in the package manual.

5.2. Examples of the basic concepts

In the following, we illustrate with some examples the basic concepts of RST and FRST.

5.2.1. Based on rough set theory

In this example, we are using the *hiring.dt* dataset [55]. It has been included in the *RoughSets* package. In order to load the data, we need to type the following command:

```
R > data(RoughSetData)
R > hiring.dt <- RoughSetData$hiring.dt
```

Firstly, we calculate the indiscernibility relation. For example, by considering the second and third attributes only, we obtain the indiscernibility relation *IND* with the command:

```
R > IND <- BC.IND.relation.RST(hiring.dt, attribute = c(2,3))
```

The *IND* object contains the equivalence classes of the objects.

After obtaining the relation, we can calculate the approximations (namely *roughset*) and positive region (namely *region*) as follows:

```
R > roughset <- BC.LU.approximation.RST(hiring.dt, IND)
R > region <- BC.positive.reg.RST(hiring.dt, roughset)
```

Furthermore, we can also construct the discernibility matrix as follows:

```
R > disc.Mat <- BC.discernibility.mat.RST(hiring.dt)
```

We can then show the results, e.g. the output of *BC.LU.approximation.RST* which is *roughset* by.

```
R > print(roughset)
```

Detailed output for these examples is available at the project web page.

5.2.2. Based on fuzzy rough set theory

This example uses the dataset *pima7.dt* which contains seven objects of the pima dataset [1]. It is included in the *RoughSets* package. Therefore, we load it with:

```
R > data(RoughSetData)
R > pima7.dt <- RoughSetData$pima7.dt
```

As in the RST case, we need to compute the indiscernibility relation, approximations and the positive region in sequential order. First, we calculate the indiscernibility relation of the considered conditional and decision attributes. For example, in this case we consider the second and third attributes only (*condAttr*) and the last column as the decision attribute (*decAttr*).

```
R > condAttr <- c(2, 3)
R > decAttr <- ncol(pima7.dt)
```

Then, we need to assign values of the *control* parameter. For the indiscernibility on the conditional attributes (the second and third attributes only), we set *lukasiewicz* and *eq.1* to be our *t*-norm and relation, respectively. Detailed descriptions of the parameters can be found in the package manual.

```
R > control.ind <- list(type.aggregation = c("t.tnorm", "lukasiewicz"),
 +   type.relation = c("tolerance", "eq.1"))
R > IND.condAttr <- BC.IND.relation.FRST(pima7.dt, attributes = condAttr,
 +   control = control.ind)
```

Since our data have nominal values in the decision attribute, we assign *crisp* for the type of aggregation and relation to calculate the indiscernibility relation on the decision attribute.

```
R > control.dec <- list(type.aggregation = "crisp", type.relation = "crisp")
R > IND.decAttr <- BC.IND.relation.FRST(pima7.dt, attributes = decAttr,
 +   control = control.dec)
```

After obtaining the relations on the conditional and decision attributes, we can calculate the lower and upper approximations as:

```
R > control <- list(t.implicator = "lukasiewicz")
R > imp.tnorm <- BC.LU.approximation.FRST(pima7.dt, IND.condAttr, IND.decAttr,
 + type.LU = "implicator.tnorm", control = control)
```

It can be seen that in this case we use *implicator.tnorm* proposed by [83] with *lukasiewicz* as the implicator. The *imp.tnorm* object is a list showing each index of objects included in lower and upper approximations based on decision concepts.

The positive region and degree of dependency can be obtained with:

```
R > region <- BC.positive.reg.FRST(pima7.dt, imp.tnorm)
```

Finally, we can construct the decision-relative discernibility matrix. For example, we use the following parameters.

```
R > control.1 <- list(type.relation = c("tolerance", "eq.1"),
 +   type.aggregation = c("t.tnorm", "min"),
 +   t.implicator = "lukasiewicz", type.LU = "implicator.tnorm")
```

The detailed explanation can be found in the manual. Then, we construct the matrix based on the *standard.red* algorithm which is based on [105].

```
R > disc.Mat <- BC.discernibility.mat.FRST(pima7.dt, type.discernibility =
 +   "standard.red",control = control.1)
```

As we mentioned above, the user can also define his or her own functions. For example, for constructing the indiscernibility relation we create the function *FUN.average* to be our aggregation operator. *FUN.average* is defined as average of all data on each considered attribute. Using the same dataset, we calculate the indiscernibility relation *IND.custom* of the second and third attributes as follows:

```
R > FUN.average <- function(data){
 + return(Reduce("+", data)/length(data))
 +}
R > control.ind <- list(type.aggregation = c("custom", FUN.average),
 +   type.relation  = c("tolerance", "eq.1"))
R > IND.custom <- BC.IND.relation.FRST(pima7.dt, attributes = condAttr,
 +   control = control.ind)
```

For calculating approximations, as in the indiscernibility relations, we can build new models. For example, we design a new model based on the OWA model by defining our own weight vectors (*w.vector*) [15].

```
R > w.vector <- matrix(0, nrow = nrow(pima7.dt))
R > m.owa <- round(nrow(pima7.dt)/2)
R > for (i in 1: m.owa){
 +   w.vector[i] <- (2^(m.owa - i))/(2^m.owa - 1)
 +}
```

Then, we assign *lukasiewicz* as the type of implicator and *t*-norm.

```
R > control <- list(t.implicator = "lukasiewicz", t.tnorm = "lukasiewicz",
 + w.owa  = w.vector)
```

We use the same dataset and the conditional and decision relations: *pima7.dt*, *IND.condAttr*, and *IND.decAttr*. Lastly, we calculate the approximation by the following command:

```
R > owa.custom <- BC.LU.approximation.FRST(pima7.dt, IND.condAttr, IND.decAttr,
 + type.LU = "owa", control)
```

To display the approximations *owa.custom*, we can execute:

```
R > print(owa.custom)


$fuzzy.lower
fuzzy.lower$1
1                2              3              4              5              6              7
1.0000000        0.3785507      0.6765217      0.3785507      0.9339130      0.9953623      1.0000000
fuzzy.lower$2
1                2              3              4              5              6              7
0.2291304        0.9026087      0.2856522      0.7055072      0.3124638      0.1307246      0.1442029
$fuzzy.upper
$fuzzy.upper$1
1                2              3              4              5              6              7
0.7708696        0.0973913      0.7143478      0.2944928      0.6875362      0.8692754      0.8557971
$fuzzy.upper$2
1                2              3              4              5
0.000000000      0.621449275    0.323478261    0.621449275    0.066086957
6                7
0.004637681      0.000000000


$type.LU
[1] "owa"
$type.model
[1] "FRST"
attr(,"class")
[1] "LowerUpperApproximation" "list"
```

It can be seen that the *owa.custom* object is a list containing the lower and upper approximations separated by the concept classes. Additionally, other descriptions, such as *type.LU*, are presented as well.

Other examples showing the use of custom functions can be seen in the package manual and on our website. The manual has been designed to show how all approaches are executed with complete related parameters.

5.3. An example for data analysis with RoughSets

In this example, we only consider implementations based on RST for prediction while examples regarding FRST can be found in the manual. The example uses a real dataset which is *wine* data included in the package.

First, we load the *wine* data from the package:

```
R > data(RoughSetData)
R > dataset <- RoughSetData$wine.dt
```

For simplicity, we divide the data into training and testing data in the following way. After shuffling the data, the training data called *wine.decTable* are 80% of all data while the rest is the testing data (*tst.wine*).

```
R > dt.Shuffled <- dataset[sample(nrow(dataset)),]
R > idx <- round(0.8 *nrow(dt.Shuffled))
R > wine.decTable <- SF.asDecisionTable(dt.Shuffled[1: idx,],
 +   decision.attr = 14, indx.nominal = 14)
R > tst.wine <- SF.asDecisionTable(dt.Shuffled[(idx + 1):nrow(dt.Shuffled),
 + -ncol(dt.Shuffled)])
R > real.val <- dt.Shuffled[(idx + 1):nrow(dt.Shuffled), ncol(dt.Shuffled),
 +   drop = FALSE]
```

Since the wine dataset contains real-valued attributes, we need to discretize them. For example, we use the *global.discernibility* method. Then we obtain cut values (called *cut.values*) by the following command:

```
R > cut.values <- D.discretization.RST(wine.decTable,
 + type.method = "global.discernibility")
```

Then we apply the cut values to the training and testing data to generate new decision tables (called *d.tra* and *d.tst*):

```
R > d.tra <- SF.applyDecTable(wine.decTable, cut.values)
R > d.tst <- SF.applyDecTable(tst.wine, cut.values)
```

Now, we have the datasets containing nominal values.

We perform feature selection using the *quickreduct.rst* method as follows:

```
R > red.rst <- FS.feature.subset.computation(d.tra, method = "quickreduct.rst")
```

A new decision table can be generated from the decision reduct *red.rst* with the following command:

```
R > fs.tra <- SF.applyDecTable(d.tra, red.rst)
```

After data preprocessing, a rule-based classifier can be induced from the training data:

```
R > rules <- RI.indiscernibilityBasedRules.RST(d.tra, red.rst)
```

It should be noted that in this case, the function needs the decision reduct as input data. Then, we can predict the testing data (*d.tst*) by considering the rules as follows:

```
R > pred.vals <- predict(rules, d.tst)
```

The predicted values *pred.vals* can be compared with the real values *real.val* as follows:

```
R > err <- 100*sum(pred.vals!=real.val)/nrow(pred.vals)
R > cat("The percentage error = ", err, "\n")

The percentage error = 8.333333
```

Detailed output for these examples and additional output is available at the project web page.

In summary, the examples illustrate the way in which the *RoughSets* package facilitates data analysis. Many options of parameter values are available to obtain good solutions. The functionality for defining our own functions as parameter values provides flexibility and opportunities for researchers to extend current models.

6. Comparison with other software packages

This section reviews some existing software libraries implementing RST and FRST. A comparison according to the functionality and capability of the software systems is shown in Table 2. In the following, we give detailed comparisons of the *RoughSets* library with four of them.

6.1. Rough Set Data Explorer (ROSE)

It is a software developed by the Laboratory of Intelligent Decision Support Systems of the Institute of Computing Science, Poznań Technical University. It is used to implement basic elements of the rough set theory and rule discovery technique [82,81]. It allows to apply the variable precision rough set defined by Ziarko [120] and the classical model by Pawlak [75] for constructing the approximations. Both ROSE and *RoughSets* allow the user to perform the basic concepts of RST. For data processing, it provides a discretization method based on an entropy measure proposed by Fayyad and Irani [22]. Feature selection is done by algorithms based on the lattice search introduced by Romański [85] and the decision-relative discernibility matrix in [92]. Some algorithms are implemented to generate rules, such as LEM2 [27,56] and the explore algorithm [65]. The software implements the L-metric or valued closeness relation based on [95] to predict new data. Even though ROSE provides more algorithms for rule induction than *RoughSets*, in the case of discretization and feature selection *RoughSets* offers more functionality.

6.2. Rough Set Exploration System (RSES)

It is a toolset for analyzing data implemented at the Group of Logic, Institute of Mathematics, University of Warsaw and the Group of Computer Science, Institute of Mathematics, University of Rzeszów, Poland [4]. It has an associated Java library called RSESlib. There are some algorithms included for handling data processing (e.g., missing value completion, discretization, feature selection, decomposition), rule induction, and classifiers based on nearest neighbor algorithms. For handling missing values, it provides four approaches: object removal, filling missing parts [31], analysis of data and treating the missing data as information. The discretization task is performed by algorithms proposed in [68]. Besides determining reducts based on exhaustive and genetic algorithms [5], it has implemented feature selection based on dynamic reducts [5,2]. Furthermore, decomposition of tables and creation of new attributes are considered in this package as well. Implementation of the covering with reducts approach [112] and LEM2 [28] is used to generate rules. Although the following classifiers are not based on rough set theory, classifiers based on nearest neighbors [21] and local transfer function [111] are implemented as well in the package. It is clear that RSES is a comprehensive software, especially for tackling data preprocessing. However, it does not provide facilities for users to operate the basic concepts of RST as in *RoughSets*.

6.3. Rough set toolkit for analysis of data (ROSETTA)

It is a software package developed by a collaboration of researchers working in the Knowledge System Group at the Norwegian University of Science and Technology (NTNU), Norway, and the Group of Logic, University of Warsaw, Poland. It is an advanced system for data analysis based on RST [72,73]. It consists of two parts which are a computational kernel and a graphical user interface (GUI) front end. Even though ROSETTA implements some algorithms as its computational kernel, it also uses the RSES library. For example, in discretization algorithms, it considers algorithms of boolean reasoning [70], entropy measure [18,23] and χ-statistics [54,58]. From a front end view, it provides a user friendly interface and allows to import datasets from external data sources, e.g., database management systems (DBMS), via the open database connectivity (ODBC) interface. Additionally, we can export results, e.g., decision rules, reducts, etc., to various formats such as XML, C++ and Prolog. In summary, the advantages of this software are that it provides a user friendly interface based on GUI and a connection with databases. *RoughSets* offers other benefits. For example, the consistency and reliability of command line facilities for batch processing and heavy experimentation. Furthermore, R as the scientific environment provides many other packages implementing the connection with databases and various file formats.

6.4. Waikato Environment for Knowledge Analysis (WEKA)

It is a popular framework for data mining tasks such as data preprocessing, classification, regression, clustering, association rules and visualization [33]. It provides a sophisticated GUI containing four menus: *Explorer*, *Experimenter*, *KnowledgeFlow*, and *SimpleCLI*. Even though there are many packages included in WEKA, in this survey we focus on a package utilizing FRST in WEKA. It was created by Jensen et al. [42]. For feature selection, it has implemented several methods, for example fuzzy QuickReduct using parameters γ [14], boundary region [48], δ [48], VQRS [11], and fuzzy discernibility matrix approach [50]. By employing search methods such as ant colony optimization (ACO) [47] and particle swarm optimization (PSO) [110], reducts can be generated. Fuzzy-rough instance selection is deployed to select qualified objects in the instance selection task [44]. Lastly, it also provides classifier methods based on nearest neighbors [43] and rule induction using hybrid fuzzy-rough rule induction and feature selection (called QuickRules) [49]. Some algorithms of feature selection, rule induction, and

Table 2
A comparison of *RoughSets* with other packages.

Components	ROSE	RSES	ROSETTA	WEKA	*RoughSets*
General views					
Theory	Rough set (RST)	RST	RST	Fuzzy rough set (FRST)	RST and FRST
Programming Language (mainly)	C++	C++ and Java	C++	Java	R
Operating System	MS Windows	MS Windows and Linux	MS Windows	MS Windows, Mac Os X, and Linux	MS Windows, Mac OS X, Linux, and Solaris
User Interface	Graphical User Interface (GUI)	GUI and scripting interface	GUI	GUI	Scripting interface
License	Use only for research, education, development, private, and non profit purposes	Freely distributed for non-commercial purposes	Use only for non-commercial purposes, a partially restricted use for algorithms of RSES library	The GNU General Public License	The GNU General Public License
Available features					
Basic concepts	Yes	No	No	No	Yes
Discretization	Yes	Yes	Yes	No	Yes
Feature selection	Yes	Yes	Yes	Yes	Yes
Instance Selection	No	No	No	Yes	Yes
Missing value completion	Yes	Yes	Yes	Yes	No
Decomposition	No	Yes	No	No	No
Rule-based classifiers	Yes	Yes	Yes	Yes	Yes
Nearest neighbor-based classifiers	No	Yes	No	Yes	Yes
Cross validation	Yes	Yes	Yes	Yes	No
Algorithms of the basic concepts					
Relations	Equivalence [75], similarity [97,98]	Equivalence [75]	Equivalence [75]	Jensen [48]	**RST:**equivalence [75]; **FRST:**Naessend [67], Jensen [48], and kernel [36]
Approximations	Pawlak [75], Ziarko [120]	Pawlak [75], Ziarko [120]	Pawlak [75], Ziarko [120]	implicator/t-norm [83], VQRS [12], OWA [15]	**RST:**Pawlak [75]; **FRST:**implicator/t-norm [83], VQRS [12], OWA [15], FVPRS [118], SFRS [35], RFRS [37], and β-PFRS [86]
Positive Region	Pawlak [75]	Pawlak [75]	Pawlak [75]	Pawlak [75]	Pawlak [75]
Discernibility Matrix	Skowron [92]	Skowron [92]	Skowron [92]	Fuzzy discernibility matrix approach [50]	**RST:** Skowron [92]; **FRST:**Tsang [105], Zhao [118], and Chen [10,9]
Algorithms of applications					
Discretization	Entroy measure [22]	Nguyen [68]	Nguyen [68], Boolean reasoning [70], entropy [18,23], and χ-statistics [54,58]	–	**RST:**maximal & local [5], global [69], and unsupervised [18]
Feature Selection	Lattice search [85], discernibility matrix [92]	Discernibility matrix [92], genetic algorithm [5], and dynamic reducts [2,5]	Discernibility matrix [92], genetic algorithm [5], and dynamic reducts [2,5]	Fuzzy QuickReduct γ [14], boundary region [48], δ [48], VQRS [11], fuzzy discernibility matrix approach [50], ACO [47], and PSO [110]	**RST:**QuickReduct [91], the greedy heuristics [113,93,41], permutation [40], discernibility matrix [92]; **FRST:**fuzzy QuickReduct [6,11,15,35,37,46,48,86,118], the near-optimal [118]
Rule Induction	LEM2 [27,56], explore [65]	Covering reducts [112] and LEM2 [28]	Covering reducts [112] and LEM2 [28]	QuickRules [49]	**RST:**indiscernibility based rules [75]; **FRST:**QuickRules [49] generalized [119]
Instance Selection	–	–	–	FRIS [44]	**FRST:**FRIS [44] and FRPS [108]
Nearest Neighbor Classifiers	–	–	–	FRNN [43,45]	**FRST:**FRNN [45], FRNN.O [88], and POSNN [107]

nearest neighbor-based classifier in the package are also implemented in *RoughSets*. However, especially for constructing the approximations, in *RoughSets* we provide not only more approaches but also custom functions for some parameters. In addition, the R environment provides the possibility to convert the *data.frame* format from/to the WEKA Attribute-Relation File Format (ARFF) by employing the *foreign* package [102].

7. Conclusions and future work

RoughSets is an R package integrating the implementations of algorithms from the rough set and fuzzy rough set theories. Its functionality allows to utilize basic concepts from both of these theories for practical data processing and analysis. It also facilitates their applications in discretization, feature selection, instance selection, rule induction, and nearest neighbor-based classifiers. It allows researchers to construct their own models based on user-defined functions while practitioners can analyze their data using the available approaches. Several usage examples have been illustrated in order to help people in getting started with the package. A comparison with other software libraries shows that *RoughSets* should be taken into account as an alternative tool for analyzing data based on the rough set theory and the fuzzy rough set theory.

In the future, we will attempt to extend our software to deal with different specific data mining problems, such as missing values [62] and imbalanced classification [61,101]. Reducing the computation time is also our concern. Additionally, we plan to improve implementations of several methods in order to make them scalable for big data [84,8].

Acknowledgements

This work was supported in part by the Spanish Ministry of Education and Science under Project TIN2011-28488, the Andalusian Research Plan P10-TIC-6858, P11-TIC-7765, and the Polish National Science Centre Grant DEC-2011/01/B/ST6/03867. Lala S. Riza would like to express his gratitude to the Department 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.

References

[1] J. Alcalá-Fdez, L. Sánchez, S. García, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández, F. Herrera, KEEL: a software tool to assess evolutionary algorithms for data mining problems, Soft Comput. 13 (3) (2009) 307–318.
[2] J. Bazan, A comparison of dynamic and non-dynamic rough set methods for extracting laws from decision tables, in: A. Skowron, L. Polkowski (Eds.), Rough Sets in Knowledge Discovery, vol. 1, Physica Verlag, Heidelberg, 1998, pp. 321–365.
[3] J.G. Bazan, M. Szczuka, RSES and RSESlib–a collection of tools for rough set computations, in: W. Ziarko, Y. Yao (Eds.), Proceedings of the 2nd International Conference on Rough Sets and Current Trends in Computing (RSCTC'2000), vol. 2005, 2000, pp. 106–113.
[4] J.G. Bazan, M. Szczuka, The rough set exploration system, in: J.F. Peters, A. Skowron (Eds.), Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 3400 LNCS, 2005, pp. 37–56.
[5] J.G. Bazan, H.S. Nguyen, S.H. Nguyen, P. Synak, J. Wróblewski, Rough set algorithms in classification problem, in: L. Polkowski, S. Tsumoto, T.Y. Lin (Eds.), Rough Set Methods and Applications, Physica-Verlag, Heidelberg, New York, 2000, pp. 49–88.
[6] R.B. Bhatt, M. Gopal, On fuzzy-rough sets approach to feature selection, Patt. Recog. Lett. 26 (2005) 965–975.
[7] Y. Caballero, R. Bello, Y. Salgado, M.M. Garca, A method to edit training set based on rough sets, Int. J. Comput. Intell. Res. 3 (2007) 219–229.
[8] C.L.P. Chen, C. Zhang, Data-intensive applications, challenges, techniques and technologies: A survey on big data, Inform. Sci. 275 (2014) 314–347.
[9] D. Chen, L. Zhang, S. Zhao, Q. Hu, P. Zhu, A novel algorithm for finding reducts with fuzzy rough sets, IEEE Trans. Fuzzy Syst. 20 (2012) 385–389.
[10] D.G. Chen, Q.H. Hu, Y.P. Yang, Parameterized attribute reduction with gaussian kernel based fuzzy rough sets, Inform. Sci. 181 (2011) 5169–5179.
[11] C. Cornelis, R. Jensen, A noise-tolerant approach to fuzzy-rough feature selection, in: Proceedings of the 2008 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2008), 2008, pp. 1598–1605.
[12] C. Cornelis, M. De Cock, A. Radzikowska, Vaguely quantified rough sets, in: Proceedings of 11th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing (RSFDGrC2007), Lecture Notes in Artificial Intelligence, vol. 4482, 2007, pp. 87–94.
[13] C. Cornelis, M. De Cock, A. Radzikowska, Fuzzy rough sets: from theory into practice, in: W. Pedrycz, A. Skowron, V. Kreinovich (Eds.), Handbook of Granular Computing, Wiley, 2008, pp. 533–552.
[14] C. Cornelis, R. Jensen, G. Hurtado, D. Ślęzak, Attribute selection with fuzzy decision reducts, Inform. Sci. 180 (2) (2010) 209–224.
[15] C. Cornelis, N. Verbiest, R. Jensen, Ordered weighted average based fuzzy rough sets, in: Proceedings of the 5th International Conference on Rough Sets and Knowledge Technology (RSKT 2010), 2010, pp. 78–85.
[16] T.M. Cover, P.E. Hart, Nearest neighbor pattern classification, IEEE Trans. Inform. Theory 13 (1967) 21–27.
[17] J. Dai, Q. Xu, Attribute selection based on information gain ratio in fuzzy rough set theory with application to tumor classification, Appl. Soft Comput. 13 (2013) 211–221.
[18] J. Dougherty, R. Kohavi, M. Sahami, Supervised and Unsupervised Discretization of Continuous Features, Morgan Kaufman, San Francisco, CA, 1995.
[19] D. Dubois, H. Prade, Rough fuzzy sets and fuzzy rough sets, Int. J. Gen. Syst. 17 (1990) 91–209.
[20] D. Dubois, H. Prade, Putting rough sets and fuzzy sets together, in: R. Słowiński (Ed.), Intelligent Decision Support Handbook of Applications and Advances of the Rough Sets Theory, Kluwer, Dordrecht, 1992, pp. 203–233.
[21] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.
[22] U.M. Fayyad, K.B. Irani, On the handling of continuous-valued attributes in decision tree generation, Mach. Learn. 8 (1992) 87–102.
[23] U.M. Fayyad, K.B. Irani, Multi-interval discretization of continuous attributes as preprocessing for classification learning, in: Proceeding Thirteenth International Joint Conference on Artificial Intelligence, Morgan Kaufman, 1995, pp. 1022–1027.
[24] E. Fix, J. Hodges, Discriminatory Analysis, Nonparametric Discrimination: Consistency Properties, Technical report, 1951.
[25] M. Forina, E. Leardi, C. Armanino, S. Lanteri, PARVUS: an extendable package of programs for data exploration, classification and correlation, J. Chem. 4 (2) (1988) 191–193.
[26] S. Greco, B. Matarazzo, R. Słowiński, Generalizing rough set theory through dominance-based rough set approach, in: Proceedings of 10th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing, part II, 2005, pp. 1–11.
[27] J.W. Grzymała-Busse, LERS – a system for learning from examples based on rough sets, in: R. Słowiński (Ed.), Intelligent Decision Support, 1992, pp. 3–18.
[28] J.W. Grzymała-Busse, A new version of the rule induction system LERS, Fund. Inform. 31 (1) (1997) 27–39.

[29] J.W. Grzymała-Busse, MLEM2: a new algorithm for rule induction from imperfect data, in: Proceedings of the 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU), 2002, pp. 243–250.
[30] J.W. Grzymała-Busse, Lers – a data mining system, in: Oded Maimon, Lior Rokach (Eds.), Data Mining and Knowledge Discovery Handbook, Springer, US, 2005, pp. 1347–1351. ISBN 978-0-387-24435-8.
[31] J.W. Grzymała-Busse, M. Hu, A comparison of several approaches to missing attribute values in data mining, in: Proceedings of the Second International Conference on Rough Sets and Current Trends in Computing RSCTC'2000, LNAI 2005, Springer-Verlag, Berlin, 2000, pp. 378–385.
[32] J.W. Grzymała-Busse, W. Rzasa, A local version of the MLEM2 algorithm for rule induction, Fund. Inform. 100 (1–4) (2010) 99–116.
[33] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA data mining software: an update, in: ACM SIGKDD Explorations Newsletter, vol. 11, 2009, pp. 10–18.
[34] D. Harrison, D.L. Rubinfeld, Hedonic prices and the demand for clean air, J. Environ. Econ. Manage. 5 (1978) 81–102.
[35] Q. Hu, S. An, D. Yu, Soft fuzzy rough sets for robust feature evaluation and selection, Inform. Sci. 180 (2010) 4384–44007.
[36] Q. Hu, D. Yu, W. Pedrycz, D. Chen, Kernelized fuzzy rough sets and their applications, IEEE Trans. Knowl. Data Eng. 23 (2011). 1649-1471.
[37] Q. Hu, L. Zhang, S. An, D. Zhang, D. Yu, On robust fuzzy rough set models, IEEE Trans. Fuzzy Syst. 20 (2012) 636–651.
[38] B. Huang, Y. Zhuang, H. Li, D. Wei, A dominance intuitionistic fuzzy-rough set approach and its applications, Appl. Math. Model. 37 (2013) 7128–7141.
[39] R. Ihaka, R. Gentleman, R: a language for data analysis and graphics, J. Comput. Graph. Statist. 5 (1996) 299–314.
[40] A. Janusz, D. Ślęzak, Rough set methods for attribute clustering and selection, Appl. Artif. Intell. 28 (3) (2014) 220–242.
[41] A. Janusz, S. Stawicki, Applications of approximate reducts to the feature selection problem, in: Proceedings of International Conference on Rough Sets and Knowledge Technology (RSKT), vol. 6954, 2011, pp. 45–50.
[42] R. Jensen, Fuzzy-Rough Data Mining with WEKA, Technical report, 2010 <http://users.aber.ac.uk/rkj/Weka.pdf>.
[43] R. Jensen, C. Cornelis, A new approach to fuzzy-rough nearest neighbour classification, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 5306 LNAI, 2008, pp. 310–319.
[44] R. Jensen, C. Cornelis, Fuzzy-rough instance selection, in: Proceedings of the 19th International Conference on Fuzzy Systems (FUZZ-IEEE 2010), 2010, pp. 1776–1782.
[45] R. Jensen, C. Cornelis, Fuzzy-rough nearest neighbour classification and prediction, Theoret. Comp. Sci. 412 (2011) 5871–5884.
[46] R. Jensen, Q. Shen, Fuzzy-rough sets for descriptive dimensionality reduction, in: Proceedings of IEEE International Conference on Fuzzy System, FUZZ-IEEE, 2002, pp. 29–34.
[47] R. Jensen, Q. Shen, Fuzzy-rough data reduction with ant colony optimization, Fuzzy Sets Syst. 149 (1) (2005) 5–20.
[48] R. Jensen, Q. Shen, New approaches to fuzzy-rough feature selection, IEEE Trans. Fuzzy Syst. 19 (2009) 824–838.
[49] R. Jensen, C. Cornelis, Q. Shen, Hybrid fuzzy-rough rule induction and feature selection, in: IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2009, pp. 1151–1156.
[50] R. Jensen, A. Tuson, Q. Shen, Extending propositional satisfiability to determine minimal fuzzy-rough reducts, in: IEEE World Congress on Computational Intelligence, WCCI 2010, 2010, pp. 1–8.
[51] R. Jensen, A. Tuson, Q. Shen, Finding rough and fuzzy-rough set reducts with SAT, Inform. Sci. 255 (2014) 100–120.
[52] J. Karami, A. Alimohammadi, T. Seifouri, Water quality analysis using a variable consistency dominance-based rough set approach, Comp., Environ. Urban Syst. 43 (2014) 25–33.
[53] J.M. Keller, M.R. Gray, J.R. Givens, A fuzzy k-nearest neighbor algorithm, IEEE Trans. Syst., Man, Cybernet. 15 (1985) 580–585.
[54] R. Kerber, ChiMerge: discretization of numeric attributes, in: AAAI-92 Proceedings of the Ninth National Conference on Artificial Intelligence, AAAI Press/MIT Press, 1992, pp. 123–128.
[55] J. Komorowski, Z. Pawlak, L. Polwski, A. Skowron, Rough sets: a tutorial, in: S.K. Pal, A. Skowron (Eds.), Rough Fuzzy Hybridization, A New Trend in Decision Making, Springer, Singapore, 1999, pp. 3–98.
[56] K. Krawiec, R. Słowiński, D. Vanderpooten, Learning of decision rules from similarity based rough approximations, in: A. Skowron, L. Polkowski (Eds.), Rough Sets in Knowledge Discovery, vol. 2, Physica Verlag, Heidelberg, 1998, pp. 37–54.
[57] C. Li, Y. Yang, M. Jia, Y. Zhang, X. Yu, C. Wang, Phylogenetic analysis of dna sequences based on k-word and rough set theory, Phys. A: Statist. Mech. Appl. 398 (2014) 162–171.
[58] H. Liu, R. Setiono, Discretization of ordinal attributes and feature selection, in: Proceedings of the Seventh International Conference on Tools with Artificial Intelligence, Washington, DC, 1995, pp. 388–391.
[59] Y. Liu, Q. Zhou, E. Rakus-Anderson, G. Bai, A fuzzy-rough sets based compact rule induction method for classifying hybrid data, in: Rough Sets and Knowledge Technology, Lecture Notes in Computer Science, vol. 7414, 2012, pp. 63–70.
[60] Y.L. Liu, Research on information technology with character pattern recognition method based on rough set theory, Advan. Mater. Res. 886 (2014) 519–523.
[61] V. López, A. Fernandez, S. García, V. Palade, F. Herrera, An insight into classification with imbalanced data: empirical results and current trends on using data intrinsic characteristics, Inform. Sci. 250 (2013) 113–141.
[62] J. Luengo, S. García, F. Herrera, On the choice of the best imputation methods for missing values considering three groups of classification methods, Knowl. Inform. Syst. 32 (1) (2012) 77–108.
[63] S. Ma, H. Liao, Y. Yuan, Intrusion detection based on rough-set attribute reduction, in: Lecture Notes in Electrical Engineering 219 LNEE, vol. 4, 2013, pp. 363–369.
[64] R.S. Michalski, A theory and methodology of inductive learning, in: R.S. Michalski, J.G. Carbonell, T.M. Mitchell (Eds.), Machine Learning, Morgan Kaufman, 1983, pp. 83–134.
[65] R. Mienko, J. Stefanowski, K. Tuomi, D. Vanderpooten, Discovery-oriented induction of decision rules, Cahier du Lamsade no. 141, 1996.
[66] R.A. Muenchen, The Popularity of Data Analysis Software, Technical report, 2013 <http://r4stats.com/articles/popularity/>.
[67] H. Naessens, H. De Meyer, B. De Baets, Algorithms for the computation of t-transitive closures, IEEE Trans. Fuzzy Syst. 10 (2002) 541–551.
[68] H.S. Nguyen, S.H. Nguyen, Discretization methods in data mining, in: A. Skowron, L. Polkowski (Eds.), Rough Sets in Knowledge Discovery, vol. 1, Physica Verlag, Heidelberg, 1998, pp. 451–482.
[69] S.H. Nguyen, On efficient handling of continuous attributes in large data bases, Fund. Inform. 48 (2001) 61–81.
[70] S.H. Nguyen, A. Skowron, Quantization of real-valued attributes, in: P.P. Wang (Ed.), Second Annual Joint Conference on Information Sciences (JCIS'95), Wrightsville Beach, North Carolina, 1995, pp. 34–37.
[71] J. Nieminen, Rough tolerance equality, Fund. Inform. 11 (3) (1988) 289–296.
[72] A. Øhrn, ROSETTA – A Rough Set Toolkit for Analysis of Data, Technical report, 2009 <http://www.lcb.uu.se/tools/rosetta/>.
[73] A. Øhrn, J. Komorowski, ROSETTA – a rough set tool kit for analysis of data, in: Proceedings of the fifth International Workshop on Rough Sets and Soft Computing (RSSC'97) at the Third Joint Conference on Information Sciences (JCIS'97), Research Triangle Park, NC, 1997, pp. 403–407.
[74] Z. Pawlak, Information systems – theoretical foundations, Inform. Sci. 6 (1981) 205–218.
[75] Z. Pawlak, Rough sets, Int. J. Comp. Sci. 11 (1982) 341–356.
[76] Z. Pawlak, Rough Sets – Theoretical Aspects of Reasoning About Data, Kluwer Academic, 1991.
[77] Z. Pawlak, A. Skowron, Rough sets and boolean reasoning, Inform. Sci. 177 (2007) 41–73.
[78] Z. Pawlak, A. Skowron, Rough sets: some extensions, Inform. Sci. 177 (2007) 28–40.
[79] Z. Pawlak, A. Skowron, Rudiments of rough sets, Inform. Sci. 177 (2007) 3–27.
[80] L. Polkowski, A. Skowron, J. Zytkow, Tolerance based rough sets, in: T.Y. Lin, A.M. Wildberger (Eds.), Soft Computing: Rough Sets, Fuzzy Logic, Neural Networks, Uncertainty Management, 1995, pp. 55–58.

[81] B. Predki, S. Wilk, Rough set based data exploration using ROSE system, in: Z.W. Ras, A. Skowron (Eds.), Foundations of Intelligent Systems, Lecture Notes in Artificial Intelligence, vol. 1609, Springer-Verlag, Berlin, 1999, pp. 172–180.
[82] B. Predki, R. Słowiński, J. Stefanowski, R. Susmaga, S. Wilk, ROSE – software implementation of the rough set theory, in: L. Polkowski, A. Skowron (Eds.), Proceedings of the Rough Sets and Current Trends in Computing'98 Conference, Lecture Notes in Artificial Intelligence, vol. 1424, Springer, Berlin, 1998, pp. 605–608.
[83] A.M. Radzikowska, E.E. Kerre, A comparative study of fuzzy rough sets, Fuzzy Sets Syst. 126 (2002) 137–156.
[84] S. Río, V. López, J.M. Benítez, F. Herrera, On the use of map reduce for imbalanced big data using random forest, Inform. Sci. 285 (2014) 112–137.
[85] S. Romanski, Operation on families of sets for exhaustive search, given a monotonic function, in: W. Beeri, C. Schmidt, N. Doyle (Eds.), Proceedings of the 3rd International Conference on Data and Knowledge Bases, 1988, pp. 310–322.
[86] J.M.F. Salido, S. Murakami, Rough set analysis of a general type of fuzzy data using transitive aggregations of fuzzy similarity relations, Fuzzy Sets Syst. 139 (2003) 635–660.
[87] M. Sarkar, Fuzzy-rough nearest neighbors algorithm, in: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, vol. 5, 2000, pp. 3556–3561.
[88] M. Sarkar, Fuzzy-rough nearest-neighbor algorithm in classification, Fuzzy Sets Syst. 158 (2007) 2123–2152.
[89] M. Sarkar, Fuzzy-rough nearest algorithms in classification, Fuzzy Sets Syst. 158 (2012) 2134–2152.
[90] C.E. Shannon, A mathematical theory of communication, Bell Syst. Tech. J. 27 (1948) 379–423. 623–656.
[91] Q. Shen, A. Chouchoulas, A modular approach to generating fuzzy rules with reduced attributes for the monitoring of complex systems, Eng. Appl. Artif. Intell. 13 (2000) 263–278.
[92] A. Skowron, C. Rauszer, The discernibility matrices and functions in information systems, in: R. Słowiński (Ed.), Intelligent Decision Support: Handbook of Applications and Advances of Rough Sets Theory, Kluwer Academic Publishers, Dordrecht, Netherland, 1992, pp. 331–362.
[93] D. Ślęzak, Approximate entropy reducts, Fund. Inform. 53 (2002) 365–390.
[94] D. Ślęzak, Approximate bayesian networks, in: B. Bounchon-Meunier, J. Gutierrez-Rios, L. Magdalena, R.R. Yager (Eds.), Technologies for Constructing Intelligent Systems: 2 Tools, Springer Verlag, 2002, pp. 313–326.
[95] R. Słowiński, J. Stefanowski, Rough set reasoning about uncertain data, Fund. Inform. 27 (2–3) (1996) 229–244.
[96] R. Słowiński, D. Vanderpooten, Similarity Relation as a Basis for Rough Approximations, Technical report, ICS Research Report 53/95, Warsaw University Technology, 1995.
[97] R. Słowiński, D. Vanderpooten, Similarity relation as a basis for rough approximations, in: P.P. Wang (Ed.), Advances in Machine Intelligence and Soft Computing, Bookwrights, Raleigh, NC, 1997, pp. 17–33.
[98] R. Słowiński, D. Vanderpooten, A generalized definition of rough approximations based on similarity, IEEE Trans. Knowl. Data Eng. 12 (2) (2000) 331–336.
[99] J. Stefanowski, On rough set based approaches to induction of decision rules, in: L. Polkowski, A. Skowron (Eds.), Rough Sets in Knowledge Discovery: 1 Methodology and Applications, Physica-Verlag, Heidelberg, 1998, pp. 500–529.
[100] J. Stefanowski, D. Vanderpooten, A general two stage approach to rule induction from examples, in: W. Ziarko (Ed.), Rough Sets, Fuzzy Sets and Knowledge Discovery, Springer-Verlag, 1994, pp. 317–325.
[101] J. Stefanowski, S. Wilk, Rough sets for handling imbalanced data: combining filtering and rule-based classifiers, Fund. Inform. 72 (2006) 379–391.
[102] R Core Team, R. Bivand, V.J. Carey, S. DebRoy, S. Eglen, R. Guha, N. Lewin-Koh, M. Myatt, B. Pfaff, B. Quistorff, F. Warmerdam, S. Weigand, Free Software Foundation Inc. foreign: Read data stored by minitab, s, sas, spss, stata, systat, weka, dbase. Technical report, 2014 <http://CRAN.R-project.org/package=foreign>. R package version 0.8-61.
[103] R Development Core Team, An Introduction to R.R Foundation for Statistical Computing, Vienna, Austria, 2008. ISBN 3-900051-12-7 <http://www.R-project.org/>.
[104] R Development Core Team, R: A language and environment for statistical computing, Technical report, R Foundation for Statistical Computing, Vienna, Austria, 2010 <http://www.r-project.org/foundation/>.
[105] E.C.C. Tsang, D.G. Chen, D.S. Yeung, X.Z. Wang, J.W.T. Lee, Attributes reduction using fuzzy rough sets, IEEE Trans. Fuzzy Syst. 16 (2008) 1130–1141.
[106] S. Tsumoto, Automated induction of medical expert system rules from clinical databases based on rough set theory, Inform. Sci. 112 (1998) 67–84.
[107] N. Verbiest, C. Cornelis, R. Jensen, Fuzzy-rough positive region based nearest neighbour classification, in: Proceedings of the 20th International Conference on Fuzzy Systems (FUZZ-IEEE 2012), 2012, pp. 1961–1967.
[108] N. Verbiest, C. Cornelis, F. Herrera, A fuzzy rough prototype selection method, Patt. Recog. 46 (2013) 2770–2782.
[109] G. Wang, Z. Zheng, Y. Zhang, RIDAS – a rough set based intelligent data analysis system, in: Proceedings of the International Conference on Machine Learning and Cybernetics, 2002, vol. 2, 2002, pp. 646–649.
[110] X. Wang, J. Yang, X. Teng, W. Xia, R. Jensen, Feature selection based on rough sets and particle swarm optimization, Patt. Recog. Lett. 28 (4) (2007) 459–471.
[111] M. Wojnarski, LTF-C: architecture, training algorithm and applications of new neural classifier, Fund. Inform. 54 (1) (2003) 89–105.
[112] J. Wróblewski, Covering with reducts – a fast algorithm for rule generation, in: Proceeding of RSCTC'98, LNAI, vol. 1424, Springer Verlag, Berlin, 1998, pp. 402–407.
[113] J. Wróblewski, Ensembles of classifiers based on approximate reducts, Fund. Inform. 47 (2001) 351–360.
[114] W.Z. Wu, W.X. Zhang, Contructive and axiomatic approaches of fuzzy approximation operators, Inform. Sci. 159 (2004) 3–4.
[115] W.Z. Wu, J.S. Mi, W.X. Zhang, Generalized fuzzy rough sets, Inform. Sci. 151 (2003) 263–282.
[116] X.D. Yu, A new patterns recognition method based on fuzzy rough sets, Appl. Mech. Mater. 380–384 (2013) 3795–3798.
[117] L.A. Zadeh, Fuzzy sets, Inform. Control 8 (1965) 338–353.
[118] S.Y. Zhao, E.C.C. Tsang, D.G. Chen, The model of fuzzy variable precision rough sets, IEEE Trans. Fuzzy Syst. 17 (2009) 451–467.
[119] S.Y. Zhao, E.C.C. Tsang, D.G. Chen, X.Z. Wang, Building a rule-based classifier – a fuzzy-rough set approach, IEEE Trans. Knowl. Data Eng. 22 (2010) 624–638.
[120] W. Ziarko, Analysis of uncertain information in the framework of variable precision rough sets, Found. Comput. Dec. Sci. 18 (3–4) (1993) 381–396.