Rseslib 3: Library of Rough Set and Machine Learning Methods with Extensible Architecture

Arkadiusz Wojna[1] and Rafał Latkowski[2]

[1] Security On-Demand, 12121 Scripps Summit Dr 320, San Diego, CA 92131, USA
[2] Loyalty Partner, Złota 59, 00-120 Warsaw, Poland
{wojna,rlatkows}@mimuw.edu.pl

Abstract. The paper presents a new generation of Rseslib library - a collection of rough set and machine learning algorithms and data structures in Java. It provides algorithms for discretization, discernibility matrix, reducts, decision rules and for other concepts of rough set theory and other data mining methods. The third version was implemented from scratch and in contrast to its predecessor it is available as a separate open-source library with API and with modular architecture aimed at high reusability and substitutability of its components. The new version can be used within Weka and with a dedicated graphical interface. Computations in Rseslib 3 can be also distributed over a network of computers.
Keywords: rough set, discernibility matrix, reduct, k nearest neighbors, machine learning, Java, Weka, distributed computing, open source.

1 Introduction

Rough set theory [20] was introduced by Pawlak as a methodology for data analysis based on approximation of concepts in information systems. Discernibility is a key concept in this methodology, which is the ability to distinguish objects, based on their attribute values. Along with theoretical research rough sets were developed in practical directions as well. To facilitate applications software tools implementing rough set concepts and methods have been developed. This paper describes one of such tools.

Rseslib 3 is a library of rough set and machine learning algorithms and data structures implemented in Java [35,36]. It is the successor of Rseslib 2 used in Rough Set Exploration System (RSES) [2]. The first version of the library started in 1993 and was implemented in C++. It was used as the core of Rosetta system [19]. Rseslib 2 was the first version of the library implemented in Java and it stands for the core of RSES. The third version of the library was entirely redesigned and all the methods available in this version were implemented from scratch. It provides algorithms for discretization, discernibility matrix, reducts, decision rules and rule-based classifiers as well as very fast implementation of the k nearest neighbors method with high accuracy distance measure and many well-known classical classification methods. The following features are distinguishing the version 3 from its predecessor:

- available as a library with an API
- open source distributed under GNU GPL license
- modular component-based architecture
- easy-to-reuse data representations and methods
- easy-to-substitute components
- available in Weka

As an open source library of rough set methods in Java Rseslib 3 fills in an uncovered gap in the spectrum of rough set software tools. The algorithms in Rseslib 3 can be used both by users who need to apply ready-to-use rough set methods in their data analysis tasks as well as by researchers interested in extension of the existing rough set methods who can use the source code of the library as the basis for their extended implementations. The library can be used also within the following external tools: Weka [8], the dedicated graphical interface Qmak and Simple Grid Manager distributing computations over a network of computers.

The library is not limited to rough sets, it contains and is open to concepts and algorithms from other areas of machine learning and data mining. That is related to another goal of the project which is to provide a universal library of highly reusable and substitutable components at a very elementary level unmet in open source data mining Java libraries available today.

The paper is organised as follows. Other software implementing rough set related methods is discussed in Section 2. The types of data handled by the library and the data related notation used in the paper are presented in Section 3. Section 4 describes all discretization methods available in Rseslib. Section 5 discusses the types of discernibility matrix and indiscernibility relations provided by the library. Section 6 defines the types of reducts available in Rseslib and describes all the implemented algorithms computing reducts. Section 7 presents the algorithms computing rules from reducts. Section 8 describes the classification models implemented in Rseslib including the reduct-based method. Section 9 enumerates other available algorithms. Section 10 introduces to modularity of the library and discusses reusability and substitutability of its components. Section 11 presents the tools that can be used with the library: Weka, the dedicated graphical interface Qmak and a tool running Rseslib-based experiments on many computers or cores. Section 12 provides examples of Rseslib usage in independent research and software projects. Section 13 concludes the paper and outlines the future plans for the project.

2 Related Work

Looking for analogous open source Java projects one can find Modlem[3] and Richard Jensen's programs[4].

[3] https://sourceforge.net/projects/modlem
[4] http://users.aber.ac.uk/rkj/?page_id=79

Modlem is a sequential covering algorithm inducing decision rules that contains some aspects of rough set theory. Numerical values are handled without discretization. Modlem as a classification method is available as Weka package.

Richard Jensen implemented a number of rough-fuzzy feature selection methods in Java. That includes a variety of search techniques, e.g. hill-climbing, ant colony optimization, genetic algorithm, as well as metrics and measures. Jensen provides also his own version of Weka with some methods included.

There are two useful libraries developed in other programming languages.

RoughSets package [23] implemented in the R programming language provides rough set and fuzzy rough models and methods. It implements the concepts of indiscernibility relations, lower and upper approximations, positive region and discernibility matrix. Using these concepts it provides the algorithms for discretization, feature selection, instance selection, rule induction, prediction and classification. RoughSets package was extended with RapidRoughSets [11] — an extension facilitating the use of the package in RapidMiner, a popular java platform for data mining, machine learning and predictive analytics.

NRough library [32] implemented in C# provides algorithms computing decision reducts, bireducts, decision reduct ensembles and decision rules. The algorithms can be used as feature selection and classification methods.

There are a number of tools providing rough set methods within graphical interface.

Rosetta [19] is the graphical tool based on the first version of Rseslib library. It provides functions for tabular data analysis supporting the overall data mining and knowledge discovery process. It provides methods computing exact and approximate reducts and generating if-then rules from computed reducts.

Rough Set Data Explorer (ROSE) [22] is the graphical tool for rough set based analysis of data. It provides methods for data processing including discretization, core and reduct computation, decision rule induction from rough approximations, and rule-based classification. As a distinctive feature ROSE includes variable precision rough set model.

Rough Set Exploration System (RSES) [2] is the graphical tool based on the second version of Rseslib library. It provides wide range of methods for data discretization, reduct compution, rule induction and rule-based classification.

3 Data

The concept of the library is based on classical representation of data in machine learning. It is assumed that a finite set of objects U, a finite set of conditional attributes $A = \{a_1, \ldots, a_n\}$ and a decision attribute dec are given. Each object $x \in U$ is represented by a vector of values $(x_1, \ldots, x_n)$. The value x_i is the value of the attribute a_i on the object x belonging to the domain of values V_i corresponding to the attribute a_i: $x_i \in V_i$. The type of a conditional attribute a_i can be either numerical, if its values are comparable and can be represented by numbers $V_i \subseteq \mathbb{R}$ (e.g.: age, temperature, height), or nominal, if its values are incomparable, i.e., if there is no linear order on V_i (e.g.: color, sex, shape).

The library contains many algorithms implementing various methods of supervised learning. These methods assume that each object $x \in U$ is assigned with a value of the decision attribute $dec(x)$ called a decision class and they learn from the objects in U a function approximating the real function dec on all objects outside U. At present the algorithms in the library assume that the domain of values of the decision attribute dec is discrete and finite: $V_{dec} = \{d_1, \ldots, d_m\}$.

The library reads data from files. Three data formats are accepted by the library:

- **ARFF**
  The format of the popular open source machine learning software WEKA [8] widely adopted in the machine learning community.
- **CSV (Comma Separated Version)**
  A popular format that can be exchanged between databases, spreadsheet programs like Microsoft Excel or Libre Office and software recognizing this format like Rseslib. To read this format Rseslib needs the description of columns called Rseslib header [36]. The header can be provided inside the file with data or in a separated file. The option of the header in a separate file enables to use the file with data by other programs without any extra conversion and eliminates the inconvenience of editing large files in case of very large data sets. Unlike in ARFF listing the values of the decision attribute is optional. The decisions can be collected directly from data if not given in the header.
- **RSES2**
  The format of RSES system.

4 Discretizations

Some algorithms require data in form of nominal attributes, e.g. some rule based algorithms like the rough set based classifier. Discretization (known also as quantization or binning) is data transformation converting data from numeric attributes into nominal attributes.

The library provides a number of discretization methods. Each method splits domain of a numerical attribute into a number of disjoint intervals. New nominal attribute is formed by encoding a numerical value into an identifier of an interval.

The discretization methods available in Rseslib are described below.

4.1 Equal Width

The range of values of a numerical attribute in a data set is divided into k intervals of equal length. The number of intervals k is the parameter of the method.

4.2 Equal Frequency

The range of values of a numerical attribute in a data set is divided into k intervals containing the same number of objects from a data set. The number of objects in particular intervals may differ by one if the size of the data set does not divide by k. The number of intervals k is the parameter of the method.

4.3 One Rule

Holte's 1R algorithm [9] tries to cut the range of values of a numerical attribute into intervals containing training objects with the same decision but it avoids very small intervals. The minimal number n of training objects that must fall into each interval is the parameter of 1R algorithm. The algorithm executes the following steps:

1. Sort the objects by the values of a numerical attribute to be discretized
2. Scan the objects in the ascending order adding them to an interval until one of the decision classes, denote it by d, has n representatives in the interval
3. While the decision of the object next in the ascending order is d add the object to the interval
4. Start the next interval as empty and go to 2

4.4 Static Entropy Minimization

Static entropy minimization [5] is a top-down local method discretizing a single numerical attribute. It starts with the whole range of values of the attribute in a data set and divides it into smaller intervals. At each step the algorithm remembers which objects from the data set fall into each interval. In a single step the algorithm searches all possible cuts in all intervals and selects the new cut c maximizing information gain, i.e. minimizing entropy:

$$E(a_i, c, S) = \frac{|S_1|}{|S|} Ent(S_1) + \frac{|S_2|}{|S|} Ent(S_2) \tag{1}$$

where

$$Ent(S) = -\sum_{j=1}^{m} \frac{|\{x \in S : dec(x) = d_j\}|}{|S|} \log\left(\frac{|\{x \in S : dec(x) = d_j\}|}{|S|}\right)$$

a_i is the attribute to be discreatized, S is the set of the objects falling into the interval on a_i containing a candidate cut c, $S_1 = \{x \in S : x_i \leq c\}$, $S_2 = \{x \in S : x_i > c\}$.

The method applies the minimum description length prinicple to decide when to stop the algorithm.

4.5 Dynamic Entropy Minimization

Dynamic entropy minimization method [5] is similar to static entropy minimization but it discretizes all numerical attributes at once. It starts with the whole set of objects and splits it into two subsets with the optimal cut selected from all numerical attributes. Then the algorithm splits each subset recursively scanning all possible cuts over all numerical attributes at each split. To select the best cut the algorithm minimizes the same formula 1 as the static method.

On average the dynamic method is faster than the static method and produces fewer cuts.

4.6 ChiMerge

ChiMerge [13] ia a bottom-up discretization method using χ^2 statistics to test whether neighbouring intervals have significantly different decision distributions. If the distributions are similar the algorithm merges the intervals into one interval. The method discretizes each numerical attribute independently.

The method has two parameters. The first parameter n is the minimal number of final intervals. The second parameter is the confidence level $(0.0 - 1.0)$ used to recognize two neighbouring intervals as different and not to merge them.

First, the algorithm calculates the threshold θ from χ^2 distribution with $m-1$ degrees of freedom and a given confidence level and starts with a separate interval for each value of a numerical attribute occuring in a data set U. At each step it merges the pair of neighbouring intervals with the minimal χ^2 value as long as this minimal value is less then θ and the number of intervals does not drop below n. χ^2 value is defined as:

$$\chi^2(S_1, S_2) = \sum_{j=1}^{m} \frac{\left(\left|S_1^j\right| - ES_1^j\right)^2}{ES_1^j} + \sum_{j=1}^{m} \frac{\left(\left|S_2^j\right| - ES_2^j\right)^2}{ES_2^j}$$

where S_1, S_2 are the sets of objects from U falling into two neighouring intervals, $S_k^j = \{x \in S_k : dec(x) = d_j\}$ and ES_k^j is the expected number of objects in S_k with the decision d_j:

$$ES_k^j = |S_k| \, \frac{|\{x \in S_1 \cup S_2 : dec(x) = d_j\}|}{|S_1 \cup S_2|}$$

4.7 Global Maximal Discernibility Heuristic

Global maximal discernibility heuristic method [17] is a top-down dynamic method discretizing all numerical attributes at once. At each step it evaluates cuts globally with respect to the whole training set. It starts with the set S^* of all pairs of objects with different decisions defined as:

$$S^* = \{\{x, y\} \subseteq U : dec(x) \neq dec(y)\}$$

At each step the algorithm finds the cut c that discerns the greatest number of pairs in the current set S^*, adds the cut c to the result set and removes all pairs discerned by the cut c from the set S^*. The optimal cut is searched among all possible cuts on all numerical attributes. The algorithm stops when the set S^* is empty.

4.8 Local Maximal Discernibility Heuristic

Local maximal discernibility heuristic method [17] selects the cuts optimizing the number of pairs of discerned objects like the global method but the procedure selecting the best cut is applied recursively to the subsets of objects obtained by splitting the data set by the proviously selected cuts.

It starts with the best cut for the whole training set U splitting it into subsets U_1 and U_2. Next the discretization algorithm selects the best cut splitting U_1 and recursively the best cuts with respect to the subsets of U_1. Next it searches independently for the best cuts for U_2. At each step the best cut is searched over all attributes.

5 Discernibility Matrix

Computation of reducts is based on the concept of discernibility matrix [27]. The library provides 4 types of discernibility matrix including types handling inconsistencies in data [21,26]. Each type is $|U| \times |U|$ matrix defined for all pairs of objects $x, y \in U$. The fields of discernibility matrix $M(x, y)$ are defined as the subsets of the set of conditional attributes: $M(x, y) \subseteq A$. If a data set contains numerical attributes discernibility matrix can be computed using either the original or the discretized numerical attributes.

The first type of discernibility matrix M^{all} depends on the values of the conditional attributes only, it does not take the decision attribute into account:

$$M^{all}(x, y) = \{a_i \in A : x_i \neq y_i\}$$

In many applications, e.g. in object classification, we want to discern objects only if they have different decisions. The second type of discernibility matrix M^{dec} discerns objects from different decision classes:

$$M^{dec}(x, y) = \begin{cases} \{a_i \in A : x_i \neq y_i\} & \text{if } dec(x) \neq dec(y) \\ \emptyset & \text{if } dec(x) = dec(y) \end{cases}$$

If data are inconsistent, i.e. if there are one or more pairs of objects with different decisions and with equal values on all conditional attributes:

$$\exists x, y \in U : \forall a_i \in A : x_i = y_i \wedge dec(x) \neq dec(y)$$

then $M^{dec}(x, y) = \emptyset$ like for pairs of objects with the same decision. To overcome this inconsistency the concept of generalized decision was introduced [21,26]:

$$\partial(x) = \{d \in V_{dec} : \exists y \in U : \forall a_i \in A : x_i = y_i \wedge dec(y) = d\}$$

If U contains inconsistent objects x, y they have the same generalized decision. The next type of discernibility matrix M^{gen} is based on generalized decision:

$$M^{gen}(x,y) = \begin{cases} \{a_i \in A : x_i \neq y_i\} & \text{if } \partial(x) \neq \partial(y) \\ \emptyset & \text{if } \partial(x) = \partial(y) \end{cases}$$

This type of discernibility matrix removes inconsistencies but discerns pairs of objects with the same original decision, e.g. an inconsistent object from a consistent object. The fourth type of discernibility matrix M^{both} discerns a pair of objects only if they have both the original and the generalized decision different:

$$M^{both}(x,y) = \begin{cases} \{a_i \in A : x_i \neq y_i\} & \text{if } \partial(x) \neq \partial(y) \wedge dec(x) \neq dec(y) \\ \emptyset & \text{if } \partial(x) = \partial(y) \vee dec(x) = dec(y) \end{cases}$$

Data can contain missing values. All types of discernibility matrix available in the library have 3 modes to handle missing values [14,15,29]:

- different value — an attribute a_i discerns x, y if the value of one of them on a_i is defined and the value of the second one is missing (missing value is treated as yet another value): $a_i \notin M(x,y) \Leftrightarrow x_i = y_i \vee (x_i = * \wedge y_i = *)$
- symmetric similarity — an attribute a_i does not discern x, y if the value of any of them on a_i is missing: $a_i \notin M(x,y) \Leftrightarrow x_i = y_i \vee x_i = * \vee y_i = *$
- nonsymmetric similarity — asymmetric discernibility relation between x and y: $a_i \notin M(x,y) \Leftrightarrow (x_i = y_i \wedge y_i \neq *) \vee x_i = *$

The first mode treating missing value as yet another value keeps indiscernibility relation transitive but the next two modes make it intransitive. Such a relation is not an equivalence relation and does not define correctly indiscernibility classes in the set U. To eliminate that problem the library provides an option to transitively close an intransitive indiscernibility relation.

6 Reducts

Reduct [27] is a key concept in rough set theory. It can be used to remove some data without loss of information or to generate decision rules.

Definition 1. *The subset of attributes $R \subseteq A$ is a (global) reduct in relation to a discernibility matrix M if each pair of objects discernible by M is discerned by at least one attribute from R and no proper subset of R holds that property:*

$$\forall x, y \in U : M(x,y) \neq \emptyset \Rightarrow R \cap M(x,y) \neq \emptyset$$

$$\forall R' \subsetneq R \, \exists x, y \in U : M(x,y) \neq \emptyset \wedge R' \cap M(x,y) = \emptyset$$

If M is a decision-dependent discernibility matrix the reducts related to M are the reducts related to the decision attribute *dec*.

Reducts defined in Definition 1 called also global reducts are sometimes too large and generate too specific rules. To overcome this problem the notion of local reducts was introduced [40].

Definition 2. *The subset of attributes $R \subseteq A$ is a local reduct in relation to a discernibility matrix M and an object $x \in U$ if each object $y \in U$ discerned from x by M is discerned from x by at least one attribute from R and no proper subset of R holds that property:*

$$\forall y \in U : M(x,y) \neq \emptyset \Rightarrow R \cap M(x,y) \neq \emptyset$$

$$\forall R' \subsetneq R \, \exists y \in U : M(x,y) \neq \emptyset \wedge R' \cap M(x,y) = \emptyset$$

It may happen that local reducts are still too large. In the extreme situation there is only one global or local reduct equal to the whole set of attributes A. In such situations partial reducts [16,18] can be helpful.

Let P be the set of all pairs of objects $x, y \in U$ discerned by a discernibility matrix M: $P = \{\{x,y\} \subseteq U : M(x,y) \neq \emptyset\}$ and let $\alpha \in (0;1)$.

Definition 3. *The subset of attributes $R \subseteq A$ is a global α-reduct in relation to a discernibility matrix M if it discerns at least $(1-\alpha)\,|P|$ pairs of objects discernible by M and no proper subset of R holds that property:*

$$|\{\{x,y\} \subseteq U : R \cap M(x,y) \neq \emptyset\}| \geq (1-\alpha)\,|P|$$

$$\forall R' \subsetneq R : |\{\{x,y\} \subseteq U : R' \cap M(x,y) \neq \emptyset\}| < (1-\alpha)\,|P|$$

Let $P(x)$ be the set of all objects $y \in U$ discerned from $x \in U$ by a discernibility matrix M: $P(x) = \{y \in U : M(x,y) \neq \emptyset\}$ and let $\alpha \in (0;1)$.

Definition 4. *The subset of attributes $R \subseteq A$ is a local α-reduct in relation to a discernibility matrix M and an object $x \in U$ if it discerns at least $(1-\alpha)\,|P(x)|$ objects discernible from x by M and no proper subset of R holds that property:*

$$|\{y \in U : R \cap M(x,y) \neq \emptyset\}| \geq (1-\alpha)\,|P(x)|$$

$$\forall R' \subsetneq R : |\{y \in U : R' \cap M(x,y) \neq \emptyset\}| < (1-\alpha)\,|P(x)|$$

The following algorithms computing reducts are available in Rseslib:

- **All Global Reducts**
  The algorithm computes all global reducts from a data set. The algorithm is based on the fact that a set of attributes is a reduct if and only if it is a prime implicant of a boolean CNF formula generated from the discernibility matrix [25]. First the algorithm calculates the discernibility matrix and then it transforms the discernibility matrix into a boolean CNF formula. Finally it applies an efficient algorithm finding all prime implicants of the formula using well-known in the field of boolean reasoning advanced techniques accelerating computations [4]. All found prime implicants are global reducts.
- **All Local Reducts**
  The algorithm computes all local reducts for each object in a data set. Like the algorithm computing global reducts it uses boolean reasoning. The first step is the same as for global reducts: the discernibility matrix specified by parameters is calculated. Next for each object x in the data set the row of the discernibility matrix corresponding to the object x is transformed into a CNF formula and all local reducts for the object x are computed with the algorithm finding prime implicants.

Dataset	Attrs	Objects	All global	All local	Global partial	Local partial
segment	19	1540	0.6	0.9	0.2	0.2
chess	36	2131	4.1	66.1	0.2	0.4
mushroom	22	5416	2.9	4.9	0.8	1.5
pendigits	16	7494	10.4	23.2	2.2	4.3
nursery	8	8640	6.5	6.7	1.5	2.8
letter	16	15000	44.6	179.7	9.7	20.5
adult	13	30162	62.1	70.1	18.0	33.0
shuttle	9	43500	91.8	92.5	22.7	48.4
covtype	12	387342	8591.9	8859.0	903.7	7173.7

Table 1. Time (in seconds) of computing decision-related reducts by Rseslib algorithms on exemplary data sets.

- **One Johnson Reduct**
  The method computes one reduct with greedy Johnson algorithm [12]. The algorithm starts with the empty set of attributes called the candidate set and adds iteratively one attribute maximizing the number of discerned pairs of objects according to the semantics of a selected discernibility matrix. It stops when all objects are discerned and checks if any of the attributes in the candidate set can be removed. The final candidate set is a reduct.
- **All Johnson Reducts**
  A version of the greedy Johnson algorithm in which the algorithm branches and traverses all possibilities rather than selecting one of them arbitrarily when more than one attribute cover the maximal number of uncovered fields of the discernibility matrix. The result is the set of the reducts found in all branches of the algorithm.
- **Global Partial Reducts**
  The algorithm finding global α-reducts described in [16]. The value α is the parameter of the algorithm.
- **Local Partial Reducts**
  The algorithm finding local α-reducts described in [16]. The value α is the parameter of the algorithm.

Algorithms computing reducts are the most time-consuming among rough set algorithms, the time cost of other steps in the overall knowledge discovery process is often negligible when compared to reduct computations. Hence it is important to provide an efficient implementation of the algorithms computing reducts. Table 1 presents the time of computing decision-related reducts by the algorithms available in Rseslib on data sets from UCI machine learning repository[5]. Numerical attributes were discretized with the local maximal discernibility method. The experiments were run on Intel Core i7-4790 3.60GHz processor.

[5] https://archive.ics.uci.edu/ml

7 Rules Generated from Reducts

Reducts described in the previous section can be used in Rseslib to generate decision rules. As reducts can be generated from a discernibility matrix using generalized decision Rseslib uses generalized decision rules:

Definition 5. *A decision rule indicates the probabilities of the decision classes at given values of some conditional attributes:*

$$a_{i_1} = v_1 \wedge \ldots \wedge a_{i_p} = v_p \Rightarrow (p_1, \ldots, p_m)$$

where p_j is defined as:

$$p_j = \frac{\left|\left\{x \in U : x_{i_1} = v_1 \wedge \ldots \wedge x_{i_p} = v_p \wedge dec(x) = d_j\right\}\right|}{\left|\left\{x \in U : x_{i_1} = v_1 \wedge \ldots \wedge x_{i_p} = v_p\right\}\right|} \tag{2}$$

A data object x is said to match a rule if the premise of the rule is satisfied by the attribute values of x: $x_{i_1} = v_1, \ldots, x_{i_p} = v_p$.

Rseslib provides new functionality regarding the semantics of missing descriptor values and missing attribute values. If rules were induced with use of a discernibility matrix then this matrix specifies similarity measure between objects (c.f. Section 5). This similarity relation is used for rule matching in such a way that a rule matches an object if the description of the rule is similar to the object with respect to the used similarity relation. In case of the different-value similarity relation implemented in the classic discernibility matrix the behaviour of rule matching is exactly as descibed above and compatibile with all other implementations not using special missing attribute value handling. If other similiarity relations and other discernibility matrices are used then different semantics of missing attribute values can be used. In such circumstances rule matching is defined according to a specified similarity relation (c.f. [15]).

Each decision rule r: $a_{i_1} = v_1 \wedge \ldots \wedge a_{i_p} = v_p \Rightarrow (p_1, \ldots, p_m)$ in Rseslib is assigned with its support in the data set U used to generate rules:

$$support(r) = \left|\left\{x \in U : x_{i_1} = v_1 \wedge \ldots x_{i_p} = v_p\right\}\right|$$

Rseslib provides two algorithms generating decision rules from reducts:

- **Rules from global reducts** (Johnson reducts are global reducts). Given a set of global reducts GR the algorithm finds all templates in the data set:

$$Templates(GR) = \left\{ \bigwedge_{a_i \in R} a_i = x_i : R \in GR, x \in U \right\}$$

  For each template the algorithm generates one rule with the decision probabilities p_j as defined in Formula 2:

$$Rules(GR) = \{t \Rightarrow (p_1, \ldots, p_m) : t \in Templates(GR)\}$$

– **Rules from local reducts**. For each object $x \in U$ the algorithm applies the selected algorithm $LR : U \mapsto \mathcal{P}(A)$ computing local reducts $LR(x)$ for x and generates the set of templates as the union of the sets of templates from all objects in U:

$$Templates(LR) = \left\{ \bigwedge_{a_i \in R} a_i = x_i : R \in LR(x), x \in U \right\}$$

The set of decision rules is obtained from the set of templates in the same way as in case of global reducts:

$$Rules(LR) = \{t \Rightarrow (p_1, \ldots, p_m) : t \in Templates(LR)\}$$

8 Classification

8.1 Rough Set Classifier

Rough set classifier provided in Rseslib uses the algorithms computing discernibility matrix, reducts and rules generated from reducts described in the previous sections. It enables to apply any of the discretization methods described in Section 4 to transform numerical attributes into nominal attributes. A user of the classifier selects a discretization method, a type of discernibility matrix and an algorithm generating reducts. The classifier computes a set of decision rules and the support of each rule in the training set.

Let *Rules* denote the computed set of decision rules. The rules are used in classification to determine a decision value when provided with an object x to be classified. First, the classifier calculates the vote of each decision class $d_j \in V_{dec}$ for the object x:

$$vote_j(x) = \sum_{\{t \Rightarrow (p_1,\ldots,p_m) \in Rules:\, x \text{ matches } t\}} p_j \cdot support(t \Rightarrow (p_1, \ldots, p_m))$$

Then the classifier assigns to x the decision with the greatest vote:

$$dec_{roughset}(x) = \max_{d_j \in V_{dec}} vote_j(x)$$

8.2 K Nearest Neighbors / RIONA

Rseslib provides an originally extended version of the k nearest neighbors (k-nn) classifier [34]. It can work with data containing both numerical and nominal attributes and implements very fast neighbor search [33] that make the classifier work in reasonable time for large data sets.

In the learning phase the algorithm induces a distance measure from a training set and constructs an indexing tree used for fast neighbor search. Optionally, the algorithm can learn the optimal number k of nearest neighbors from the training set. The distance measure is the weighted sum of distances between

Classifier	Search method	Training time (sec)	Classification time (sec)	Accuracy
Weka IBk	linear search	0.63	2674.86	93.9%
Weka IBk	KDTree	120.32	20.54	93.9%
Rseslib KNN	Rseslib KNN	27.64	3.06	96.5%

Table 2. Comparison of the nearest neighbor ($k = 1$) search methods from Rseslib and Weka for the *covtype* dataset (387342 training instances, 193670 test instances). The BallTree and CoverTree search methods available for Weka IBk were also used in the test but they failed reporting errors.

values of two objects on all conditional attributes. The classifier provides two metrics for nominal attributes: Hamming metric and Value Difference Metric (VDM), and three metrics for numerical attributes: the city-block Manhattan metric, Interpolated Value Difference Metric (IVDM) and Density-Based Value Difference Metric (DBVDM). IVDM and DBVDM metrics are adaptations of VDM metric to numerical attributes. For computation of the weights in the distance measure three methods are available: distance-based method, accuracy-based method and a method using perceptron.

While classifying an object the classifier finds k nearest neighbors in the training set according to the induced distance measure and it applies one of three methods of voting for the decision by the found neighbors: equally weighted, with inverse distance weights or with inverse square distance weights.

The classifier has also the mode to work as RIONA algorithm [6]. This mode implements a classifier combining the k-nn method with rule induction where the nearest neighbors not validated by additional rules are excluded from voting.

K nereast neighbors method in Rseslib implements very fast nearest neighbors search algorithm based on center-based indexing of training instances and using double criterion to prune searching. Table 2 presents time comparison between 1-nn search methods from Rseslib and Weka on an exemplary large data set. The training time of the Rseslib method is over 4 times shorter than the training time of the Weka method and the classification time is almost 7 times shorter. It is worth mentioning that at the same time the distance measure induced by the Rseslib method gives a significantly higher classification accuracy than the distance measure of the Weka method.

8.3 K Nearest Neighbors with Local Metric Induction

K nearest neighbors with local metric induction is the k nearest neighbors method extended with an extra step - the classifier computes a local metric for each classified object [28]. While classifying an object, first the classifier finds a large set of the nearest neighbors (according to a global metric). Then it generates a new, local metric from this large set of neighbors. At last, the k nearest neighbors are selected from this larger set of neighbors according to the locally induced metric and used to vote for the decision.

In comparison to the standard k-nn algorithm this method improves classification accuracy particularly for the case of data with nominal attributes. It is reasonable to use this method rather for large data sets (2000 training objects or more).

8.4 Classical Classifiers

Rseslib delivers also implementations of classifiers well-known in the machine learning community (see [36] for more details):

- **C4.5** - decision tree developed by Quinlan
- **AQ15** - rule-based classifier with a covering algorithm
- **Neural network** - classical backpropagation algorithm
- **Naive Bayes** - simple Bayesian network
- **SVM** - support vector machine
- **PCA** - classifier using principal component analysis
- **Local PCA** - classifier using local principal component analysis
- **Bagging** - metaclassifier combining a number of "weak" classifiers
- **AdaBoost** - another popular metaclassifier

9 Other Algorithms

Beside rough set and classification methods Rseslib provides many other machine learning and data mining algorithms. Each algorithm is available as separate class or method and easy to use as an independent component. That includes:

- **Data transformation:** missing value completion (non-invasive data imputation by Gediga and Duentsch), attribute selection, numerical attribute scaling, new attributes (radial, linear and arithmetic transformations)
- **Data filtering:** missing values filter, Wilson's editing, Minimal Consistent Subset (MSC) by Dasarathy, universal boolean function based filter
- **Data sampling:** with repetitions, without repetitions, with given class distribution
- **Data clustering:** k approximate centers algorithm
- **Data sorting:** attribute value related, distance related
- **Rule induction:** from global reducts, from local reducts, AQ15 algorithm
- **Metric induction:** Hamming and Value Difference Metric (VDM) for nominal attributes, city-block Manhattan, Interpolated Value Difference Metric (IVDM) and Density-Based Value Difference Metric (DBVDM) for numerical attributes, attribute weighting (distance-based, accuracy-based, perceptron)
- **Principal Component Analysis (PCA):** OjaRLS algorithm
- **Boolean reasoning:** two different algorithms generating prime implicant from a CNF boolean formula
- **Genetic algorithm scheme:** a user provides cross-over operation, mutation operation and fitness function only
- **Classifier evaluation:** single train-and-classify test, cross-validation, multiple test with random train-and-classify split, multiple cross-validation (all types of tests can be executed on many classifiers)

10 Extensible Modular Component-Based Architecture

Providing a collection of rough set and machine learning algorithms is not the only goal of Rseslib. It is designed also to assure maximum reusability and substitutability of the existing components in new components of the library. Hence a strong emphasis is put on its modularity. The code is separated into loosely related elements as small as possible so that each element can be used independently of other elements. For each group of the elements of the same type a standardizing interface is defined so that each element used in an algorithm can be easily substituted by any other element of the same type. Code separation and standardization is applied both to the algorithms and to the objects.

The previous sections presented the range of algorithms available in Rseslib. Below there is a list of the objects in the library implementing various data-related mathematical concepts that can be used as isolated components:

- **Basic:** attribute, data header, data object, boolean data object, numbered data object, data table, nominal attribute histogram, numeric attribute histogram, decision distribution
- **Boolean functions/operators:** attribute value equality, numerical attribute interval, nominal attribute value subset, binary discrimination, metric cube, negation, conjunction, disjunction
- **Real functions/operators:** scaling, perceptron, radius function, multiplication, addition
- **Integer functions:** discrimination (discretization, 3-value cut)
- **Decision distribution functions:** nominal value to decision distribution, numeric value to vicinity-based decision distribution, numeric value to interpolated decision distribution
- **Vector space:** vector, linear subspace, principal components subspace, vector function
- **Linear order**
- **Indiscernibility relations**
- **Distance measures:** Hamming, Value Difference Metric, city-block Manhattan, Interpolated Value Difference Metric, Density-Based Value Difference Metric, metric-based indexing tree
- **Rules:** boolean function based rule, equality descriptors rule, partial matching rule
- **Probability:** gaussian kernel function, hypercube kernel function, m-estimate

The structure of rough set algorithms in Rseslib is one of the examples of the component-based architecture (see Figure 1). Each of the six modules: *Discretization, Logic, Discernibility, Reducts, Rules* and *Rough Set Classifier* provides well-abstracted algorithms with clearly defined interfaces that allow algorithms from other modules to use them as their components. For example, the algorithms computing reducts from the *Reducts* module use a discernibility matrix from the *Discernibility* module and one of the methods computing prime implicants of a CNF boolean formula from the *Logic* module. It is easy to extend

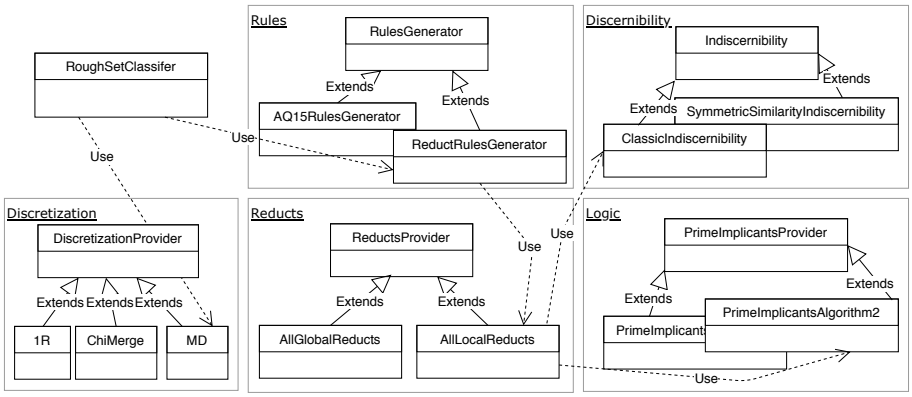

Fig. 1. Examples of relations between Rseslib modules containing rough set algorithms on simplified UML diagram

each module with implementation of a new method and to add the new method as an alternative in all components using the module.

The component-based architecture of Rseslib makes it possible to implement unconventional combinations of data mining methods. For example, perceptron learning is used as one of the attribute weighting methods in the algorithm computing a distance measure between data objects. Estimation of value probability at given decision is another example of such combination: it uses k nearest neighbors voting as one of the methods defining conditional value probability.

11 Tools

11.1 Rseslib classifiers in Weka

Weka [8] is a non-commercial suite of open source machine learning and data mining software written in Java. It is one of the most popular platforms used by data scientists and researchers for data analysis and predictive modeling with downloads counted in millions per year. It provides four graphical interfaces and one command line interface for its users. Weka has the system of external packages updated independently of the core of Weka that allows people all over the world to contribute to Weka and maintain easily their Weka extensions. Such extensions can be easily dowloaded and installed in each Weka installation.

Rseslib is such an official Weka package available from Weka repository. Rseslib version 3.1.2 (the latest at the moment of preparing this paper) provides three Rseslib classifiers with full configuration in Weka:

- Rough set classifier
- K nearest neighbours / RIONA
- K nearest neighbours with local metric induction

These three classifiers can be used, tested and compared with other classifiers within all Weka interfaces.

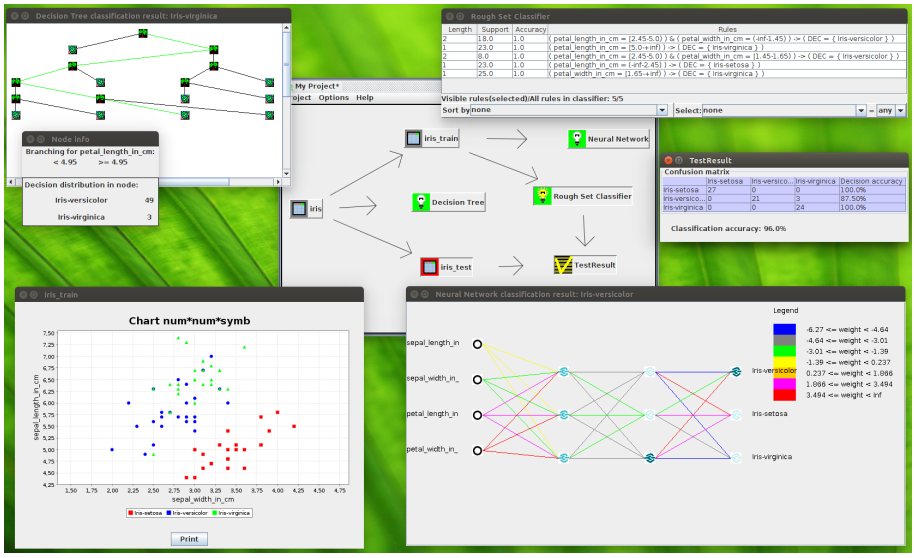

Fig. 2. Graphical user interface Qmak dedicated to Rseslib library

11.2 Graphical Interface Qmak

Qmak is a graphical user interface dedicated to Rseslib library (see Figure 2). It is a tool for data analysis, data classification, classifier evaluation and interaction with classifiers. Qmak provides the following features:

- visualization of data, classifiers and single object classification
- interactive classifier modification by a user
- classification of test data with presentation of misclassified objects
- experiments on many classifiers: single train-and-classify test, cross-validation, multiple test with random train-and-classify split, multiple cross-validation

Qmak 1.0.0 (the latest at the moment of preparing this paper) with Rseslib version 3.1.2 provides visualization of 5 classifiers: rough set classifier, k nearest neighbors, C4.5 decision tree, neural network and principal component analysis classifier. Visualization of a rough set classifier presents the decision rules of the classifier (see Figure 3). The rules can be filtered and sorted by attribute occurrence, attribute values, length, support and accuracy. Visualization of classification by rough set classifier shows the decision rules matching a classified object enabling the same types of filtering and sorting criteria as visualization of the classifier.

Users can implement new classifiers and their visualization and add them easily to Qmak. It does not require any change in Qmak itself. A new classifier can be added using GUI or in the configuration file.

Qmak is available from Rseslib homepage. Help on Qmak can be found in the main menu of the application.

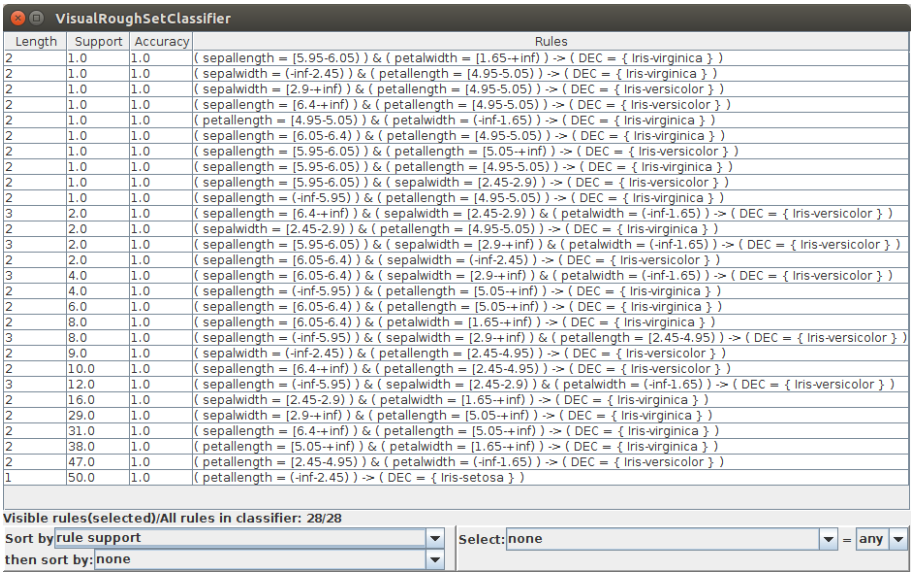

Fig. 3. Visualization of the rough set classifier presenting the rules computed from the *iris* data set sorted by rule support

11.3 Computing in Cluster

Simple Grid Manager is a tool for running massive Rseslib-based experiments on all available computers. It allows to create ad-hoc cluster with no prior configuration or additional cluster resource manager. SGM is the successor of DIXER — the previous version of software dedicated to Rseslib 2 [3]. Using SGM a user can create an ad-hoc cluster of computers by running the server module on one machine and the client module on all machines assigned to run the experiments (the elements of the cluster, see Figure 4).

The server reads experiment lists from script files, distributes tasks between all available client machines, collects results of executed tasks and stores them in a result file. The main features of the tool are:

- Executes train-and-test experiments with any set of classifiers from Rseslib library (or user written classifiers compatible with Rseslib standards)
- Allows ad-hoc cluster creation without any configuration and maintenance
- Automatically resumes failed jobs and skips completed jobs in case of restart
- Uses robust communication and allows relying by nodes launched on NAT/Firewall knots
- Enables utilizing multi-core architectures by executing many client instances on one machine

In order to use Simple Grid Manager a user needs only to create a list of experiments in a text file. Each experiment is specified with the name of a

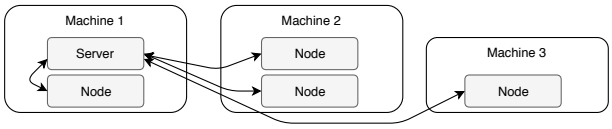

Fig. 4. Simpe Grid Manager allows running experiments on many computers by creating ad-hoc cluster with no prerequsites on cluster configuration.

classifier class, training data file name, test data file name and a list of options for the classifier. SGM executes the experiments on the cluster in such way that each experiment is executed on one node. The classification results are stored in the result file. SGM can execute experiments with any class (i.e. also user-written) that extends the interface *Classifier*.

Simple Grid Manager is available from Rseslib homepage. The guide on how to run the distributed experiments can be found in [36].

12 Rseslib Usage Examples

Rseslib has been successfully used in various independent research projects and software tools.

Hu [10] used the rough set classifier from Rseslib to build a classifier that uses a decision table obtained by pairwise comparisons between training objects based on the preference relation from the PROMETHEE method.

Adamczyk [1] used the rough set classifier to evaluate the candidate attribute sets generated in subsequent populations in the parallel algorithm for feature selection based on asynchronous particle swarm optimization.

K nearest neighbors method from Rseslib was successfully applied to environmental sound recognition [7] that can be used, for example, in an intruder detection system. The Rseslib classifier gave the best accuracy winning with 8 other Weka classifiers including Support Vector Machine and Random Forest. The classifiers were tested with 8 different sets of features, the classification accuracy of the method from Rseslib was close to perfect (in the range 99.6% - 99.79%) regardless of the number of features used in the tests. Moreover, the best accuracy of K-NN was not paid by the computational time - the classifier ranked also as one of the fastest among the tested methods.

The effectiveness of the Rseslib classifier in sound recognition was confirmed by another study conducted for the problem of context awareness of a service robot [24]. K-NN method was selected among 8 classification algorithms as one of the two methods giving satisfactory classification accuracy. Further tests were carried out to find the optimal parameter values of the K-NN method for the problem of context awareness of a service robot [30].

K nearest neighbors method with local metric induction from Rseslib was applied to the problem of liver cancer diagnosis and compared with the IBkLG method from Weka and the K-NN method from Rseslib [31]. K-NN with local

metric induction gave the best accuracy 98.8% and the best recall 99.3% among the three tested lazy classifiers.

Rseslib, along with Weka and RapidMiner, is supported also as the programming framework on two platforms used to run and test data mining and machine learning algorithms. The first one, Debellor, is an open source platform with stream-oriented architecture for scalable data mining [37,38]. The second one, TunedIT, is the platform for automated evaluation, benchmarking and comparison of machine learning algorithms [39]. In particular, the existing Rseslib algorithms can be run and tested both on Debellor and on TunedIT.

Rseslib was used also as the programming framework for discretization and computation of reducts in mahout-extensions[6], a library extending Mahout with atttribute selection methods. Mahout is an extensible programming environment and framework for building scalable algorithms in machine learning. The mahout-extensions library uses also some algorithms from Rseslib, e.g. the algorithm computing a discernibility matrix and the ChiMerge discretization method.

13 Conclusions and Future Work

The paper presents the contents of Rseslib 3 library that is designed to be used both by users who need to apply ready-to-use rough set or other data mining methods in their data analysis tasks as well as by researchers interested in extension of the existing methods. More information on Rseslib 3 and its tools can be found on the home page[7] and in the user guide [36].

The development of Rseslib 3 is continued. The repository of the library[8] is maintained by GitHub and is open to new contributions from all researchers and developers willing to extend the library. There is ongoing work on a classifier specialized in imbalanced data. The algorithms computing reducts are planned to be added to Weka package as attribute selection methods. Discretizations are also to be added to Weka package as separate algorithms. We are going to add Rseslib to Maven repository and to investigate the possibility of connecting Rseslib to RapidMiner.

Acknowledgment. We would like to thank Professor Andrzej Skowron for his support and mentorship over the project and for his advice on the development and Professor Dominik Ślęzak for his remarks to this paper. It must be emphasized that the library is the result of joint effort of many people and we express our gratitude to all the contributors: Jan Bazan, Rafał Falkowski, Grzegorz Góra, Wiktor Gromniak, Marcin Jałmużna, Łukasz Kosson, Łukasz Kowalski, Michał Kurzydłowski, Łukasz Ligowski, Michał Mikołajczyk, Krzysztof Niemkiewicz, Dariusz Ogórek, Marcin Piliszczuk, Maciej Próchniak, Jakub Sakowicz, Sebastian Stawicki, Cezary Tkaczyk, Witold Wojtyra, Damian Wójcik and Beata Zielosko.

[6] https://github.com/wgromniak/mahout-extensions
[7] http://rseslib.mimuw.edu.pl
[8] https://github.com/awojna/Rseslib

References

1. Adamczyk, M.: Parallel feature selection algorithm based on rough sets and particle swarm optimization. In: Proceedings of the 2014 Federated Conference on Computer Science and Information System. ACSIS, vol. 2, pp. 43–50 (2014)
2. Bazan, J.G., Szczuka, M.: The rough set exploration system. LNCS Transactions on Rough Sets III 3400, 37–56 (2005)
3. Bazan, J.G., Latkowski, R., Szczuka, M.: DIXER - distributed executor for rough set exploration system. In: Ślęzak, D., Yao, J., Peters, J., Ziarko, W., Hu, X. (eds.) Proceedings of the 10th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing. LNCS, vol. 3642, pp. 39–47. Springer (2005)
4. Brown, F.M.: Boolean Reasoning: The Logic of Boolean Equations. Kluwer Academic Publishers, Dordrecht (1990)
5. Fayyad, U., Irani, K.: Multi-interval discretization of continuous-valued attributes for classification learning. In: Proceedings of the 13th International Joint Conference on Artificial Intelligence. pp. 1022–1027. Morgan Kaufmann (1993)
6. Góra, G., Wojna, A.: RIONA: a new classification system combining rule induction and instance-based learning. Fundamenta Informaticae 51(4), 369–390 (2002)
7. Grama, L., Rusu, C.: Choosing an accurate number of mel frequency cepstral coefficients for audio classification purpose. In: Proceedings of the 10th International Symposium on Image and Signal Processing and Analysis. pp. 225–230. IEEE (2017)
8. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witen, I.: The weka data mining software: An update. SIGKDD Explorations 11(1), 10–18 (2009)
9. Holte, R.C.: Very simple classification rules perform well on most commonly used datasets. Machine learning 11(1), 63–90 (1993)
10. Hu, Y.C.: Rough sets for pattern classification using pairwise-comparison-based tables. Applied Mathematical Modelling 37(12-13), 7330–7337 (2013)
11. Janusz, A., Stawicki, S., Szczuka, M., Ślęzak, D.: Rough set tools for practical data exploration. In: Proceedings of the 10th International Conference on Rough Sets and Knowledge Technology. LNCS, vol. 9436, pp. 77–86. Springer (2015)
12. Johnson, D.S.: Approximation algorithms for combinatorial problems. Journal of computer and system sciences 9(3), 256–278 (1974)
13. Kerber, R.: Chimerge: Discretization of numeric attributes. In: Proceedings of the 10th National Conference on Artificial Intelligence. pp. 123–128. Aaai Press (1992)
14. Kryszkiewicz, M.: Properties of incomplete information systems in the framework of rough sets. In: Polkowski, L., Skowron, A. (eds.) Rough Sets in Knowledge Discovery 1: Methodology and Applications. pp. 422–450. Physica-Verlag (1998)
15. Latkowski, R.: Flexible indiscernibility relations for missing attribute values. Fundamenta Informaticae 67(1-3), 131–147 (2005)
16. Moshkov, M., Piliszczuk, M., Zielosko, B.: Partial covers, reducts and decision rules in rough sets: Theory and applications. Studies in Computational Intelligence 145 (2008)
17. Nguyen, H.S.: Discretization of Real Value Attributes: A Boolean Reasoning Approach. Ph.D. thesis, Warsaw University (1997)
18. Nguyen, H.S., Ślęzak, D.: Approximate reducts and association rules - correspondence and complexity results. In: Zhong, N., Skowron, A., Ohsuga, S. (eds.) Proceedings of the International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing. LNCS, vol. 1711, pp. 137–145. Springer (1999)

19. Øhrn, A., Komorowski, J., Skowron, A., Synak, P.: The design and implementation of a knowledge discovery toolkit based on rough sets - the rosetta system. In: Polkowski, L., Skowron, A. (eds.) Rough Sets in Knowledge Discovery 2: Applications, Case Studies and Software Systems, pp. 376–399. Physica-Verlag (1998)
20. Pawlak, Z.: Rough Sets - Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordrecht (1991)
21. Pawlak, Z., Skowron, A.: Rudiments of rough sets. Information sciences 177(1), 3–27 (2007)
22. Prędki, B., Wilk, S.: Rough set based data exploration using rose system. In: Raś, Z.W., Skowron, A. (eds.) Foundations of Intelligent Systems, LNCS, vol. 1609, pp. 172–180. Springer-Verlag, Berlin (1999)
23. Riza, L.S., Janusz, A., Bergmeir, C., Cornelis, C., Herrera, F., Ślęzak, D., Benitez, J.M.: Implementing algorithms of rough set theory and fuzzy rough set theory in the R package "RoughSets". Information sciences 287, 68–89 (2014)
24. Rusu, C., Grama, L.: Recent developments in acoustical signal classification for monitoring. In: Proceedings of the 5th International Symposium on Electrical and Electronics Engineering. IEEE (2017)
25. Skowron, A.: Boolean reasoning for decision rules generation. In: Komorowski, J., Raś, Z.W. (eds.) Proceedings of the 7th International Symposium on Methodologies for Intelligent Systems. LNCS, vol. 689, pp. 295–305. Springer (1993)
26. Skowron, A., Grzymała-Busse, J.W.: From rough set theory to evidence theory. In: Yager, R.R., Kacprzyk, J., Fedrizzi, M. (eds.) Advances in the Dempster-Shafer Theory of Evidence, pp. 193–236. Wiley, New York (1994)
27. Skowron, A., Rauszer, C.: The discernibility matrices and functions in information systems. In: Slowinski, R. (ed.) Intelligent Decision Support, Handbook of Applications and Advances of the Rough Sets Theory, pp. 331–362. Kluwer Academic Publishers, Dordrecht (1992)
28. Skowron, A., Wojna, A.: K nearest neighbors classification with local induction of the simple value difference metric. In: Proceedings of the 4th International Conference on Rough Sets and Current Trends in Computing. LNCS, vol. 3066, pp. 229–234. Springer-Verlag (2004)
29. Stefanowski, J., Tsoukiàs, A.: On the extension of rough sets under incomplete information. In: Zhong, N., Skowron, A., Ohsuga, S. (eds.) New Directions in Rough Sets, Data Mining, and Granular-Soft Computing, 7th International Workshop, RSFDGrC '99, Yamaguchi, Japan, November 9-11, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1711, pp. 73–81. Springer (1999)
30. Telembici, T., Grama, L.: Detecting indoor sound events. Acta Technica Napocensis - Electronics and Telecommunications 59(2), 13–17 (2018)
31. Tiwari, M., Chakrabarti, P., Chakrabarti, T.: Performance analysis and error evaluation towards the liver cancer diagnosis using lazy classifiers for ilpd. In: Proceedings of the 2nd International Conference on Soft Computing Systems. CCIS, vol. 837, pp. 161–168. Springer (2018)
32. Widz, S.: Introducing NRough framework. In: Proceedings of the International Joint Conference on Rough Sets. LNCS, vol. 10314, pp. 669–689. Springer (2017)
33. Wojna, A.: Center-based indexing for nearest neighbors search. In: Proceedings of the 3rd IEEE International Conference on Data Mining. pp. 681–684. IEEE Computer Society Press (2003)
34. Wojna, A.: Analogy-based reasoning in classifier construction (phd thesis). LNCS Transactions on Rough Sets IV 3700, 277–374 (2005)

35. Wojna, A., Latkowski, R.: Rseslib 3: Open source library of rough set and machine learning methods. In: Proceedings of the International Joint Conference on Rough Sets. LNCS, vol. 11103, pp. 162–176. Springer (2018)
36. Wojna, A., Latkowski, R., Kowalski, Ł.: RSESLIB: User Guide, http://rseslib.mimuw.edu.pl/rseslib.pdf
37. Wojnarski, M.: Debellor: A data mining platform with stream architecture. LNCS Transactions on Rough Sets IX 5390, 405–427 (2008)
38. Wojnarski, M.: Debellor: Open source modular platform for scalable data mining. In: Proceedings of the 17th International Conference on Intelligent Information Systems (2009)
39. Wojnarski, M., Stawicki, S., Wojnarowski, P.: Tunedit.org: System for automated evaluation of algorithms in repeatable experiments. In: Proceedings of the 7th International Conference on Rough Sets and Current Trends in Computing. LNCS, vol. 6086, pp. 229–234. Springer-Verlag (2010)
40. Wróblewski, J.: Covering with reducts - a fast algorithm for rule generation. In: Proceedings of the 1st International Conference on Rough Sets and Current Trends in Computing. LNCS, vol. 1424, pp. 402–407. Springer-Verlag (1998)