Rseslib 3: Open Source Library of Rough Set and Machine Learning Methods

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.

1 Introduction

Rough set theory [15] 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. 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 [14]. 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. 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 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 [1], 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.

Looking for analogous open source Java projects one can find Modlem[3] and Richard Jensen's programs[4]. Modlem is a Weka package providing a covering algorithm inducing decision rules. The algorithm contains some aspects of rough set theory. Richard Jensen developed a number of programs in Java providing various rough set methods, some of them are provided with their source code.

There are useful libraries of rough set methods developed in other programming languages: RoughSets [18] in R and NRough [23] in C#. RoughSets package was extended with RapidRoughSets [8] - an extension facilitating the use of the package in RapidMiner, a popular java platform for data mining, machine learning and predictive analytics. There are a number of tools providing rough set methods within graphical interface like RSES [2], Rosetta [14] or ROSE [17].

2 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 in three formats: ARFF, CSV and RSES2.

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

3 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 following discretization methods are available in Rseslib:

- Equal width intervals
- Equal frequency intervals
- Holte's 1R algorithm [7]
- Entropy minimization (static and dynamic) [5]
- ChiMerge algorithm [10]
- Maximal discernibility (MD) heuristic (global and local) [13]

4 Discernibility Matrix

Computation of reducts is based on the concept of discernibility matrix [21]. The library provides 4 types of discernibility matrix. Each type is $|U| \times |U|$ matrix defined for all pairs of objects $x, y \in U$. The values 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 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 [16,20]:

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

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 [11]:

- 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.

5 Reducts

Reduct [21] 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 [26].

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 [12] 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 [19]. 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.

- **One Johnson Reduct**
  The method computes one reduct with greedy Johnson algorithm [9]. 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 [12]. The value α is the parameter of the algorithm.
- **Local Partial Reducts**
  The algorithm finding local α-reducts described in [12]. The value α is the parameter of the algorithm.

The table below presents time (in seconds) of computing decision-related reducts by particular algorithms on some data sets. Numerical attributes were discretized with the local maximal discernibility method. The experiments were run on Intel Core i7-4790 3.60GHz processor.

Dataset	Attributes	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

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

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 the option to allow the values v_k in the descriptors of a rule to be missing values: $a_{i_k} = *$. An object x satisfies a descriptor with missing value $a_{i_k} = *$ if the value of the attribute a_{i_k} on x is missing: $x_{i_k} = *$.

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 Definition 5:

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

- **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:

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

7 Classification

7.1 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 listed in Section 3 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:

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

7.2 K Nearest Neighbors / RIONA

Rseslib provides an originally extended version of the k nearest neighbors (k-nn) classifier [24]. It can work with data containing both numerical and nominal attributes and implements fast neighbor search 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 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 algorithm 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.

7.3 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 [22]. 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).

7.4 Classical Classifiers

Rseslib delivers also implementations of classifiers well-known in the machine learning community (see [25] 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
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

8 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: discretizations, 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)

9 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, 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. 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. It is easy to extend 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.

10 Tools

10.1 Rseslib classifiers in Weka

Weka [1] is a very popular machine learning and data mining software equipped with the system of packages updated independently of Weka core allowing people all over the world to contribute to Weka and maintain easily their extensions.

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 neighbors / RIONA and k nearest neighbors with local metric induction. These three classifiers can be used, tested and compared with other classifiers within all Weka interfaces.

10.2 Graphical Interface Qmak

Qmak is a graphical user interface dedicated to Rseslib library. 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 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 1). 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.

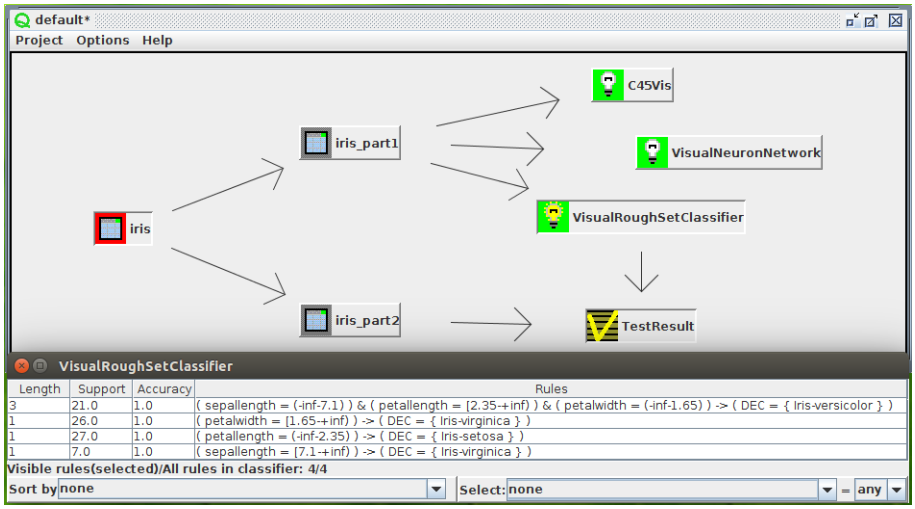

Fig. 1. Qmak project panel with instance of rough set classifier displayed

10.3 Computing in Cluster

Simple Grid Manager is a tool for running massive Rseslib-based experiments on all available computers. It is the successor of the previous version of software dedicated to Rseslib 2 [3]. Using SGM a user can create an ad-hoc cluster of computers by running server part on one machine and client part on all machines designated to run the experiments. 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 that allows creation of a cluster over non-reliable networks
- Enables utilizing multi-core architectures by executing many client instances on one machine

Simple Grid Manager is available from Rseslib homepage. The guide on how to run the distributed experiments can be found in [25].

11 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[5] and in the user guide [25].

The development of Rseslib 3 is continued. The repository of the library[6] 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 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.

References

1. Weka 3: Data mining software in java. http://www.cs.waikato.ac.nz/ml/weka
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. Holte, R.C.: Very simple classification rules perform well on most commonly used datasets. Machine learning 11(1), 63–90 (1993)
8. 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)
9. Johnson, D.S.: Approximation algorithms for combinatorial problems. Journal of computer and system sciences 9(3), 256–278 (1974)

[5] http://rseslib.mimuw.edu.pl
[6] https://github.com/awojna/Rseslib

10. Kerber, R.: Chimerge: Discretization of numeric attributes. In: Proceedings of the 10th National Conference on Artificial Intelligence. pp. 123–128. Aaai Press (1992)
11. Latkowski, R.: Flexible indiscernibility relations for missing attribute values. Fundamenta Informaticae 67(1-3), 131–147 (2005)
12. Moshkov, M., Piliszczuk, M., Zielosko, B.: Partial covers, reducts and decision rules in rough sets: Theory and applications. Studies in Computational Intelligence 145 (2008)
13. Nguyen, H.S.: Discretization of Real Value Attributes: A Boolean Reasoning Approach. Ph.D. thesis, Warsaw University (1997)
14. 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)
15. Pawlak, Z.: Rough Sets - Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordrecht (1991)
16. Pawlak, Z., Skowron, A.: Rudiments of rough sets. Information sciences 177(1), 3–27 (2007)
17. 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)
18. 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)
19. 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)
20. 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)
21. 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)
22. 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)
23. Widz, S.: Introducing NRough framework. In: Proceedings of the International Joint Conference on Rough Sets. LNCS, vol. 10314, pp. 669–689. Springer (2017)
24. Wojna, A.: Analogy-based reasoning in classifier construction (phd thesis). LNCS Transactions on Rough Sets IV 3700, 277–374 (2005)
25. Wojna, A., Latkowski, R., Kowalski, Ł.: RSESLIB: User Guide, http://rseslib.mimuw.edu.pl/rseslib.pdf
26. 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)