STUDIA INFORMATICA 2005
Volume 26 Number 4 (65)

Marek SIKORA, Aleksandra GRUCA
Politechnika Śląska, Instytut Informatyki
Robert GRUCA
Mentor Graphics Polska sp. z o.o.

TRS LIBRARY – TOOL FOR INDUCING AND POSTPROCESSING OF DECISION RULES

Summary. This paper presents TRS library, which implements decision rules induction algorithms based on tolerance rough sets model. Algorithms used for filtering and approximating rules proposed by their authors are significant part of the library. Until now the library was available as a binary file or using command line interpreter. Recently graphical user interface was created to simplify experiments and analysis.

Keywords: decision rules, filtering rules, generalizing rules, classification

BIBLIOTEKA TRS – NARZĘDZIE DO INDUKCJI I POSTPROCESSINGU REGUŁ DECYZYJNYCH

Streszczenie. W artykule opisano bibliotekę TRS, udostępniającą indukcję reguł decyzyjnych za pomocą tolerancyjnego modelu zbiorów przybliżonych. Ważną częścią biblioteki są zaproponowane przez jej autorów algorytmy, umożliwiające filtrację i uogólnianie wyznaczonych reguł. Dotychczas korzystanie z biblioteki było możliwe za pośrednictwem wersji skompilowanej lub poprzez opracowany interpreter skryptów, obecnie opracowano również interfejs użytkownika, dzięki czemu łatwiej można przeprowadzać eksperymenty i analizy.

Słowa kluczowe: reguły decyzyjne, filtracja reguł, uogólnianie reguł, klasyfikacja

1. Introduction

Rough sets theory [8] provides algorithms to analyze data tables where records are described by a vector of features. Particularly, if one or more attributes of this vector are decision attributes it is possible to generate a set of decision rules. Generated rules represent

data patterns and can be used for description (knowledge discovery) or classification. Standard rough sets model requires discretization of numerical attributes. Among many standard rough sets model generalizations, the most interesting one is tolerance model [15]. This model does not require discretization of numerical attributes, however tolerance threshold has to be defined for each attribute except the decision one. Tolerance threshold decides whether two (or more) objects are similar or not in view of a certain attribute (or a set of attributes).

Among all applications implementing rough sets theory for data analysis one can find DataLogic, Rossetta [7], RSES[1], LERS[3] and RSL library. All of them provide standard rough sets model based rules induction and post processing algorithms.

This paper describes the library which can analyze data using tolerance-based rough sets. Apart from standard algorithms the library implements algorithms for evaluating quality of induced rules and their generalization and filtration. These methods have been proposed by the authors [11] to reduce number of rules used for description and classification. Additionally two heuristic rules induction algorithms have been implemented – RMatrix algorithm [13] and MODLEM [14] algorithm with additional rule quality measure parameter [12].

Following chapters present most important algorithms implemented in TRS library and describe library communication application structure.

2. TRS Library

TRS library was created as a tool for experiments on reducing number of rules used for describing decision classes. Sets of rules generated on the basis of standard rough sets theory are very large what makes it quite difficult to adapt this theory in knowledge discovery in databases. For this reason, the TRS library provides many heuristic methods of reducing the number of generated rules. Basic algorithms implemented in the library are described below. Because of their extensive documentation the algorithms are presented only in general. Some of them are considered as standard tools that can be found in literature and other methods, proposed by the authors, have been published among others in [10, 11, 12, 13].

2.1. Algorithms of searching tolerance thresholds vector

Let $\boldsymbol{DT}=(U,A\cup\{d\})$ be a decision table, where U is a set of objects, A is a set of conditional attributes, and d is a decision attribute. A domain of an attribute a is denoted by D_a. The $\tau \subseteq U\times U$ relation is similarity relation. For each attribute $a\in A$ there is determined a distance function $\delta_a: D_a\times D_a\rightarrow[0,\infty]$ of the following properties $\forall x,y\in U\ \delta_a(a(x),a(x))=0$ and

$\delta_a(a(x),a(y))=\delta_a(a(y),a(x))$, where $a(x)$ means the value of attribute $a\in A$ for $x\in U$. TRS library has two distance functions implemented – known as *diff* and *vdm* [15].

With a fixed subset $B\subseteq A$ of conditional attributes for each attribute $a_i\in B,\ i=1,2,..,n$, a value ε_{a_i} called tolerance threshold is set. Relation $\tau_B(\varepsilon_{a_1},\varepsilon_{a_2},...,\varepsilon_{a_n})$ is defined in the following way:

$$\forall x,y\in U \quad <x,y>\in \tau_B(\varepsilon_{a_1},\varepsilon_{a_2},..,\varepsilon_{a_n}) \Leftrightarrow \forall a_i\in B[\,\delta_{a_i}(a_i(x),a_i(y))\le \varepsilon_{a_i}\,] \tag{1}$$

The set of objects similar to the object $x\in U$ regarding the attribute set $B\subseteq A$ determines uncertainty function value $I_B:U\to 2^U$ which is defined in the following way:

$$\forall y\in U \quad y\in I_B(x) \Leftrightarrow <x,y>\in \tau_B(\varepsilon_{a_1},\varepsilon_{a_2},...,\varepsilon_{a_n}) \tag{2}$$

In other words we can say that $I_B(x)$ set is a tolerance set of the element x.

When tolerance thresholds are determined it is possible to determine a set of decision rules [15] that create descriptions of decision classes. As a decision class we mean the set $X_v=\{x\in U: d(x)=v\}$, where $v\in D_d$.

A single decision rule has the following form

$$(a_1,V_{a_1})\wedge ..\wedge (a_k,V_{a_k})\to (\mathrm{d,v}) \tag{3}$$

where each (a,V_a) is called descriptor, $V_a\subseteq D_a$, and v is a value of decision class attribute.

Each rule is constructed on the base of the object-generator $x\in U$. A range V_a of descriptor (a,V_a) occurring in the rule in form (3), is determined on the base of value of tolerance threshold ε_a in a following way:

if the attribute a is of nominal type, then:

$$V_a=\{a(y)\in D_a: y\in I_a(x)\} \tag{4}$$

if a is of continuous type, then:

$$V_a=[\,v_{min}\,;\,v_{max}\,] \text{ where } v_{\min}=\min_{y\in U}\{a(y)\in D_a : y\in I_a(x)\} \tag{5}$$

and $v_{\max}=\max_{y\in U}\{a(y)\in D_a : y\in I_a(x)\}$.

Accuracy and generality (coverage) of obtained rules is determined on the base of distance measure δ_a and tolerance thresholds' values ε_{a_i}. One can select the form of the δ_a function arbitrarily, however selection of the ε_{a_i} thresholds is not obvious.

If there are decision table $\boldsymbol{DT}=(U,A\cup\{d\})$ and distance function δ_a for each $a\in A$ defined, the new decision table $\boldsymbol{DT'}=(U',A'\cup\{D\})$ can be created, where:

$U'=\{<x_i,y_j>: (<x_i,y_j>\in U\times U)\wedge (i\le j)\}$,

$A'=\{a':U'\to \boldsymbol{R}^+ : a'(<x,y>)=\delta_a(a(x),a(y))\}$,

$$D(<x,y>)=\begin{cases}0 & for \quad d(x)=d(y)\\ 1 & otherwise.\end{cases}$$

It can be easily noticed, that sorting objects from U' ascending according to values of any $a' \in A'$, one receives every possible value for which the power of set $I_a(x)$, $x \in U$ varies. These values are all reasonable ε_a values that should be considered during tolerance thresholds searching. Choosing ε_a for each $a \in A$ one receives a tolerance thresholds vector. Because the set of all possible vectors is very large $(\frac{1}{2}\prod_{a \in A}(card(D_a)^2 - card(D_a)) + 1)$ a genetic algorithm was implemented in TRS library for searching good tolerance thresholds vector. Binary representation of a single specimen (thresholds vector) was assumed and so-called block positional notation with standardization [4] was used. Each attribute had number of bits necessary to encode all numbers of tolerance thresholds acceptable for that attribute assigned.

The following function was used as a tolerance thresholds vector evaluation criterion [15]:

$$w\ \gamma(d) + (1-w)v_{SRI}(R_d, R_{I_A}) \tag{6}$$

where $\gamma_B(d) = \frac{card(POS_B(d))}{card(U)}$, $POS_B(d)$ is a B-positive region [8] of the decision table $\boldsymbol{DT}=(U, A \cup \{d\})$, $R_d=\{<x,y> \in U \times U : d(x)=d(y)\}$, $R_{I_A} = \{< x, y > \in U \times U: y \in I_A(x)\}$, $0<w\leq 1$ and $v_{SRI}(X,Y)$ is a standard rough inclusion [15].

The function mentioned above is used for searching for tolerance thresholds values for conditional attributes, that as many objects of the same decision attribute value as possible stay in mutual relation, concurrently limiting to minimum cases in which the relation concerns objects of different values of the decision attribute.

In the TRS library rule quality evaluation measures [2] were adapted as a tolerance thresholds vector evaluation criterion (specimen adaptation in the genetic algorithm). These measures allow evaluating rule accuracy and generality (coverage) [2], preferring rules of big accuracy and coverage. One can easily notice that analogous relationship exists in case of searching good tolerance thresholds. The authors are searching for tolerance threshold values for which all possible objects of the same decision attribute value stay in mutual relation. Concurrently tolerance threshold values are supposed to minimize the number of cases for which the relation concerns objects of different values of decision attribute. Details of rules adaptation and genetic algorithm mechanism are described in [9].

Algorithm of searching the vector of identical tolerance threshold value for each attribute is the other algorithm implemented in TRS library. The requested vector has the form $(\varepsilon,..,\varepsilon)$. As the initial (minimal) value of ε we take 0 and a final (maximal) value is ε=1. Additionally the parameter k is defined to increment ε on each iteration (usually k=0.1). For each vector, from (0,..,0) to (1,…1), the tolerance thresholds vector evaluation criterion is calculated (it is possible to use identical evaluation criteria as for genetic algorithm).

Apart from searching tolerance thresholds algorithms, the TRS library also implements relative reducts searching algorithms (based on the generalized discernibility matrix modulo d [15]), computation decision table core and values of attributes significance factor [15].

2.2. Rule quality measures

Rules describing decision classes (sets of objects of the same decision attribute value) should both represent general dependencies in data and be easy to interpret. It is possible when the number of rules for the decision class is small and rules describing it are strong considering two basic criteria: accuracy and coverage. The most important features characterizing the rule are accuracy (dependency described by the rule should be valid in most of the cases) and coverage (the rule should recognize as many objects as possible).

Rules matching both conditions are considered to be high quality rules. There is a high probability that these rules reflect general regularity that appears in the data.

To find the best rules the rule quality evaluation criteria are needed. Usually measures based on contingency matrix created for each rule are used.

We determine the contingency table describing the rule $\varphi \rightarrow \psi$ behavior as the following square matrix:

$n_{\varphi\psi}$	$n_{\varphi\neg\psi}$	n_{φ}
$n_{\neg\varphi\psi}$	$n_{\neg\varphi\neg\psi}$	$n_{\neg\varphi}$
n_{ψ}	$n_{\neg\psi}$	

where:

$n_{\varphi} = n_{\varphi\psi} + n_{\varphi\neg\psi} = card(U_{\varphi})$ denotes the number of objects that are recognized by the rule $\varphi \rightarrow \psi$,

$n_{\neg\varphi} = n_{\neg\varphi\psi} + n_{\neg\varphi\neg\psi} = card(U_{\neg\varphi})$ denotes the number of objects that are not recognized by the rule $\varphi \rightarrow \psi$,

$n_{\psi} = n_{\varphi\psi} + n_{\neg\varphi\psi} = card(U_{\psi})$ denotes the number of objects that belong to the decision class described by the rule $\varphi \rightarrow \psi$,

$n_{\neg\psi} = n_{\varphi\neg\psi} + n_{\neg\varphi\neg\psi} = card(U_{\neg\psi})$ denotes the number of objects that do not belong to the decision class described by the rule $\varphi \rightarrow \psi$,

$n_{\varphi\psi} = card(U_{\varphi} \cap U_{\psi})$ denotes the number of objects that support the rule $\varphi \rightarrow \psi$,

$n_{\varphi\neg\psi} = card(U_{\varphi} \cap U_{\neg\psi})$; $n_{\neg\varphi\psi} = card(U_{\neg\varphi} \cap U_{\psi})$; $n_{\neg\varphi\neg\psi} = card(U_{\neg\varphi} \cap U_{\neg\psi})$.

Using information from the contingency table and the fact, that for any known rule we know $card(U_{\psi})$ and $card(U_{\neg\psi})$ values, we can express each measure used in TRS library as a function of $n_{\varphi\psi}$, $n_{\varphi\neg\psi}$.

Among many rules quality measures the most important are accuracy and coverage:

$$q^{accuracy}_{\varphi\rightarrow\psi}((<n_{\varphi\psi},n_{\varphi\rightarrow\psi}>)=\frac{n_{\varphi\psi}}{n_{\varphi}} \quad , \quad q^{coverage}_{\varphi\rightarrow\psi}((<n_{\varphi\psi},n_{\varphi\rightarrow\psi}>)=\frac{n_{\varphi\psi}}{n_{\psi}} \tag{7}$$

Other rules quality measures are trying to combine accuracy and coverage. Besides accuracy and coverage the following measures have also been implemented in TRS library: *Brazdil, Cohen, Coleman, Coverage, Gain, IKIB, IREP, Michalski, Pearson.* Detailed descriptions of above measures are among others in [2, 10].

In the library, there is also a possibility to compute above measures values for objects uniquely covered by the rule. Moreover values of mixed measure described by the following formula:

$$q(r)_{new}=\frac{q(r)_{unique}}{q(r)_{normal}} \tag{8}$$

can be calculated, where $q(r)_{new}$ is the new measure value, $q(r)_{unique}$ is the measure q value for objects which uniquely recognize rule r , $q(r)_{normal}$ is the typical calculated measure value.

The number of conditional descriptors contained by the rule is very important for interpretation of the dependence represented by the rule. For the decision rule $\varphi\rightarrow\psi$, the set $descr(\varphi\rightarrow\psi)$ contains all attributes creating conditional descriptors in this rule. The set $descr(\varphi\rightarrow\psi)$ is defined in the following way:

$descr(\varphi\rightarrow\psi)=\{a:$ (a,V) is the conditional descriptor in rule $\varphi\rightarrow\psi\}$

The formula proposed in TRS library that evaluate the rule quality in accordance to number of conditional descriptors $q^{length}:RUL\rightarrow[0,1)$ is defined as:

$$q^{length}(r)=1-\frac{card(descr(r))}{card(A)} \tag{9}$$

There is also possible to evaluate rule quality in view of both quality measure and length: $q^{function\_name}_{\varphi\rightarrow\psi}\ q^{length}(\varphi\rightarrow\psi)$, where $q^{functio\_name}_{\varphi\rightarrow\psi}$ can be any of rule quality measure mentioned above.

Especially the formula $q^{accuracy}_{\varphi\rightarrow\psi}\ q^{length}(\varphi\rightarrow\psi)$ permits to evaluate both: rule's $\varphi\rightarrow\psi$ accuracy and complexity.

2.3. Rules induction algorithms

When the tolerance thresholds vector is determined (particularly it could be a zero vector) the decision rules can be determined. TRS library implements standard rules induction algorithm from so-called relative reducts [8]. Three versions of this algorithm are available: induction of all rules, induction of fixed number of rules, rules induction from quasi-minimal relative reduct (the heuristic Johnson [6] algorithm is used). The algorithm uses generalized

discernibility matrix modulo d [15], what limits analyzed data set to several thousands objects. RMatrix algorithm, proposed by TRS library's authors, is also based on discernibility matrix [11, 13]. This algorithm is shortly described below:

Let $\boldsymbol{DT}=(U,A\cup\{d\})$ be a decision table, where $U=\{x_1,x_2,..,x_n\}$. Generalized discernibility matrix modulo d for $\boldsymbol{DT}$ is a square matrix $M_d(\boldsymbol{DT})=\{c_{i,j}: 1\le i,j\le n\}$ of elements defined as follows:

$$\begin{cases} c_{ij}=\{a\in A:(\delta(x_i,y_i)>\varepsilon_a)\wedge(d(x_i)\neq d(y_j))\}, \\ c_{ij}=\{\varnothing:(d(x_i)=d(y_j))\} \end{cases} \tag{10}$$

When $x_i,y_j\in U$ have the same value of decision attribute, there is an empty set in discernibility matrix c_{ij} cell. In the other case, there is a set of attributes which values are sufficient to distinguish object x_i from x_j.

RMatrix algorithm uses information included in the discernibility matrix modulo d. Each rule is built on the basis of a certain generator-object. Each object matches one row (column) in the matrix $M_d(\boldsymbol{DT})$. The attribute most frequently appearing in the row allows distinguishing in the best way objects from decision classes other than generator's decision class among generator-objects. Calculating a rank of attributes considering their appearance frequency in the appropriate row of the matrix $M_d(\boldsymbol{DT})$ one can obtain a quasi-minimal relative reduct [6] for the generator-object.

The RMatrix algorithm implemented in the library generates one rule from each generator-object. Adding another descriptor makes the rule more accurate, simultaneously limiting rule's coverage. The rule quality evaluation measure makes output rule not too fitting to training data.

The RMatrix algorithm can use one of known evaluation measures (*accuracy*, *Michalski*, *Brazdil*, *IREP*, *Pearson*, *Cohen*, *Coleman, IKIB, gain* [2, 10]).

Apart from the RMatrix algorithm, TRS library implements also MODLEM [14] algorithm which does not require prior discretization of numerical data. To reduce the number of rules generated on the basis of MODLEM algorithm, proposed modification consists in, alike in RMatrix algorithm, the evaluation of the created rule (after each algorithm iteration) using rule quality measure. As the quality measure value decreases, the rule forming process stops. This modification allows generating less, but more general, rules that are characterized by good description and classification. Detailed description of MODLEM algorithm can be found in [14] and the modification mentioned above in [12].

RMatrix Algorithm

input: *decision table* ***DT***$=(U, A\cup\{d\})$, *tolerance thresholds vector* $(\varepsilon_{a_1},\varepsilon_{a_2},..,\varepsilon_{a_m})$, *q – rule quality evaluation measure, x-object rule generator, an order of conditional attributes* $(a_{i_1},a_{i_2},..,a_{i_m})$ *so as the most frequently appearing in* c_x *attribute is the first (attribute appearing the most rarely is the last)*

begin

create rule r, which has the decision descriptor (d, d(x)) only; $r_{best} := r$;

for each $j := 1, \ldots, m$

add descriptor $(a_{i_j},V_{a_{i_j}})$ *to the conditional part of rule r where*

$V_{a_{i_j}} = \{a_{i_j}(y) \in X_{a_{i_j}} : y \in I_{a_{i_j}}(x)\}$

if $q_p(r) > q_p(r_{best})$ ***then*** $r_{best} := r$

return r_{best}

2.4. Rules generalization and filtration

Determining decision rules on the basis of tolerance rough sets we are aware of the fact that obtained rules may occur to be approximate ones. Regardless of generating all the minimal decision rules or generating rules on the basis of heuristic algorithm, the set of obtained rules is large, what limits their description abilities (in data mining context). TRS library implements several algorithms for so-called rules post processing. Post processing is used to limit the number of rules (increasing their description abilities) simultaneously keeping their good classification abilities.

In TRS library post processing can be performed in two ways:

- rules generalization (rules shortening, rules joining)
- rules filtration (removing rules from the set that we don't need in view of some criterion)

The process of shortening consists in removing some descriptors from the conditional part of the rule. Removing descriptor makes rule less accurate but increases its coverage. Each unshortened rule has the value of quality measure assigned. Shortening is being processed as long as the measure value is not less than some level defined by the user (particularly the user may demand shorten rules to have identical quality). This parameter is defined individually for each decision class. Descriptors deleting order is set in accordance to hill-climbing strategy [5].

Rules joining consists in obtaining single but more general rule from two rules that are less general. Joining algorithm implemented in the library is based on following assumptions:

- only rules describing the same decision class can be joined

- we will try to join rules $\varphi 1 \rightarrow \psi$ and $\varphi 2 \rightarrow \psi$. If $descr(\varphi 1 \rightarrow \psi) \subseteq descr(\varphi 2 \rightarrow \psi)$ or $descr(\varphi 2 \rightarrow \psi) \subset descr(\varphi 1 \rightarrow \psi)$. We will join only rules that their conditional parts are build on basis of similar set of conditional attributes
- joining consists in combining the set of equivalent conditional descriptors values. If the descriptor (a, V_a^1) appears in the conditional part of the rule $\varphi_1 \rightarrow \psi$ and the descriptor (a, V_a^2) appears in the conditional part of the rule $\varphi_2 \rightarrow \psi$, then after joining those descriptors there will be created the descriptor (a, V_a) with the properties $V_a^1 \subseteq V_a$ and $V_a^2 \subseteq V_a$.

Controlling the rules joining process is proceeded on the basis of the following rules:

- from two rules r_1 and r_2 with the properties $q_{r_1} > q_{r_2}$ the rule r_1 is chosen as the "base" on the basis of which a new joined rule will be created. The notation q_{r_1} denotes the value of the rule r_1 quality evaluation measures;
- conditional descriptors are joined sequentially, the order of descriptors joining depends on the value of the function which evaluates a newly created rule r (q_r). In order to point the descriptor which is the best for joining the hill-climbing strategy is used.
- the process of joining is finished when the new rule r recognizes all positive training examples recognized by rules r_1 and r_2;
- if r is the new rule and $q_r \geq \lambda$, then in the place of rules r_1 and r_2 the rule r is inserted into the decision class description; otherwise rules r_1, and r_2 cannot be joined. The parameter λ defines the limit value, and the rules quality by a given quality evaluation measure cannot go below this value. In particular, before beginning the joining process one can create a table which contains "initial" values of evaluation functions and afterwards use the table as the table of threshold values for rules joining; then for all joined rules r_1 and r_2, $\lambda = \max\{q_{r_1}, q_{r_2}\}$.

Filtration of the decision rules set consists in removing from descriptions of decision classes the rules that are unwanted according to the established criterion. Removed rules are considered as weak rules (i.e. too detailed, recognized only by some generator-objects, in other words they fit too much to data) or unnecessary for classification. Reducing descriptions of decision classes, filtering does not interfere with the structure of obtained rules, but only limits their number.

In TRS library two types of filtration algorithms are implemented:

- not considering classification accuracy of an unfiltered rules set.
- considering classification accuracy of not filtered rules set.

First approach is represented by filtration algorithm “From coverage”. First the rules ranking is determined, and then building of coverage training set is started from the best rule. Next rules are being added according to their ranking order. When all examples from training set are covered, remaining rules (not added yet) are rejected. One of the algorithm parameter is the *Rule* parameter which determines if, after adding the rule to output set, the values of quality measures for other rules should be evaluated. Evaluation, if necessary, should be executed in attitude to the remaining (not covered yet) set of examples.

Second approach to filtration is represented by „Forward” and „Backward” algorithms. Both of them, except rules ranking created by the selected rule quality measure, use information about classification accuracy obtained by all rules. To preserve filtration independence of test data, during filtration process the classification is processed on so-called tuning set, which is the set of objects independent from train set (*File* parameter in the algorithms).

For “Forward” filtration the initial description of each decision class consists of a single best rule. Then, for description of each class a single rule is added. If accuracy of the class increases, the rule remains in the description, otherwise another rule is considered. The order of considered rules is set in accordance to rules ranking determined by the rule quality measure. Adding rules to description of decision class is stopped when the set reaches the same classification accuracy as the rules set not filtered.

„Backward” filtration algorithm is based on the opposite idea. From each decision class the rules are removed, beginning from the weakest rules. The order of rules removing is set in accordance to rules ranking determined by the rule quality measure. Keeping the difference in the accuracy between the most and the less accurate decision class ensures that filtered decision rules set has the same sensitivity as unfiltered one.

Additionally TRS library implements a simple algorithm to remove rules of quality less then the user-defined threshold.

All algorithms mentioned above can use the quality evaluation measures mentioned in Chapter 2.2. Filtration algorithms are described among others in [10].

2.5. Classification

Classification is the process of assigning objects to their corresponding decision classes only on the basis of knowledge of features characterizing them.

Classification algorithm is the automatic decision algorithm. The algorithm automatically assigns values of decision attributes on the basis of the conditional attributes values and the set of determined decision rules.

Efficiency of the decision algorithm is measured by its classification accuracy. By the classification accuracy we mean the ratio of correctly assigned objects to all test objects.

Objects recognized by rules of one decision class we assume as objects classified to this class. When the object is recognized by rules belonging to descriptions of different decision classes, the value of the decision attribute cannot be assigned unambiguously. For such classification TRS library uses the so-called voting mechanism [5]. Each obtained rule has a certain level of confidence assigned (simply, this is the rule quality measure value). In TRS library classification is proceeded as the aggregation of rules confidence levels of each decision class recognizing the test object. The object is assigned to the class that has the maximal sum value. Sometimes it happens that none of rules from obtained decision classes descriptions recognizes the test object. In that case it is possible to evaluate the distance from the test object to any rule and to assume, that rules close enough to the test object recognize it.

To summarize, TRS library implements the following scheme of assigning decision attribute's value to the test object. For each test object *u* there is determined the confidence level to every decision class according to the formula:

$$conf(X_v, u) = \sum_{r \in RUL_{X_v}(\mathrm{DT}), dist(r,u) \le \varepsilon} (1\text{-}dist(r,u))\, quality(r) \tag{11}$$

where $dist(r,u)$ is the distance from the test object u to the rule r (i.e. Euclidean, Hamming), ε is maximal acceptable distance from the object to the rule (particularly when ε=0, the classification is made by rules that accurately recognize the test object), quality(*r*) value is the strength of each rule's vote (the confidence level mentioned above). Finally, the decision function *f* assigning decision attribute value to any object u is defined by the following formula:

$$f(u) = v_{max} \Leftrightarrow conf(X_{v_{max}}, u) = \max_{v \in D_d} (conf(X_v, u)) \tag{12}$$

3. Automatic script generation application structure

TRS library graphical user interface is the application based on tabbed sheets structure. Main window is split into several tabs (fig. 1) grouped thematically: preparing data (including data loading from database or file), tolerance thresholds searching, rules determining, filtration, etc. On each tab sheet specific algorithm parameters can be set.

Script tab (fig. 2) allows preparing the script in two different ways – generating complete script automatically or adding certain commands one-by-one. The checkbox group placed in the upper left corner of the main window can be used to generate script consisting of selected (checked) commands. The algorithms parameters are being read from corresponding tabs while generating script. Another method of preparing customized script is adding selected

commands one-by-one using context menu (i.e. using two different algorithms to generate rules is possible).

Each command placed in the script begins with its code i.e. *GenerateRules* and ends with *EndCommand.* Between them algorithm parameters are set. For example determining rules by RMatrix algorithm with quality measure *Pearson* causes adding the following lines to the script:

GenerateRules
Algorithm *FromRMatrix*
Quality *Pearson*
QualityParam *0*
Coverage *No*
EndCommand

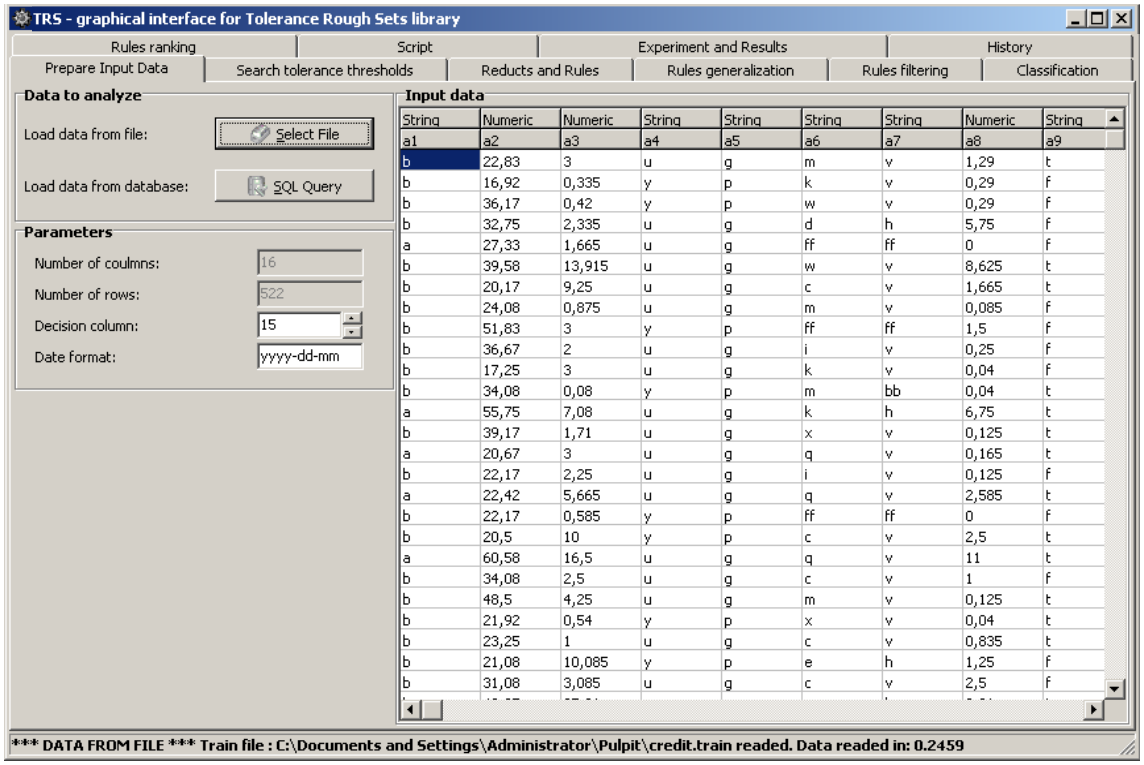

Fig. 1. Main window of TRS library graphical user interface
Rys. 1. Główne okno graficznego interfesju użytkownika biblioteki TRS

After script generation user selects some testing method for the loaded data set. Three methods are available: single experiment based on train and test files, 5-fold or 10-fold cross validation (the most popular testing methods). In the case of cross validation the program generates sets of random records for train and test and eventually tune files, and automatically performs series of 5 or 10 experiments.

All created sets are saved in a directory created in the location indicated by the user.

The application can load data from database (using ODBC mechanism) or text file with the specified data format:

Cols: *Attrubutes_number*
Rows: *Objects_number*
DecAttr: *Decision_attribute_number*
DateFormat: *Data_format*
Attribute1_type *Attribute2_type ....* *Attributen_type*
Attribute1_name *Attribute2_name ...* *Attributen_name*
……data……

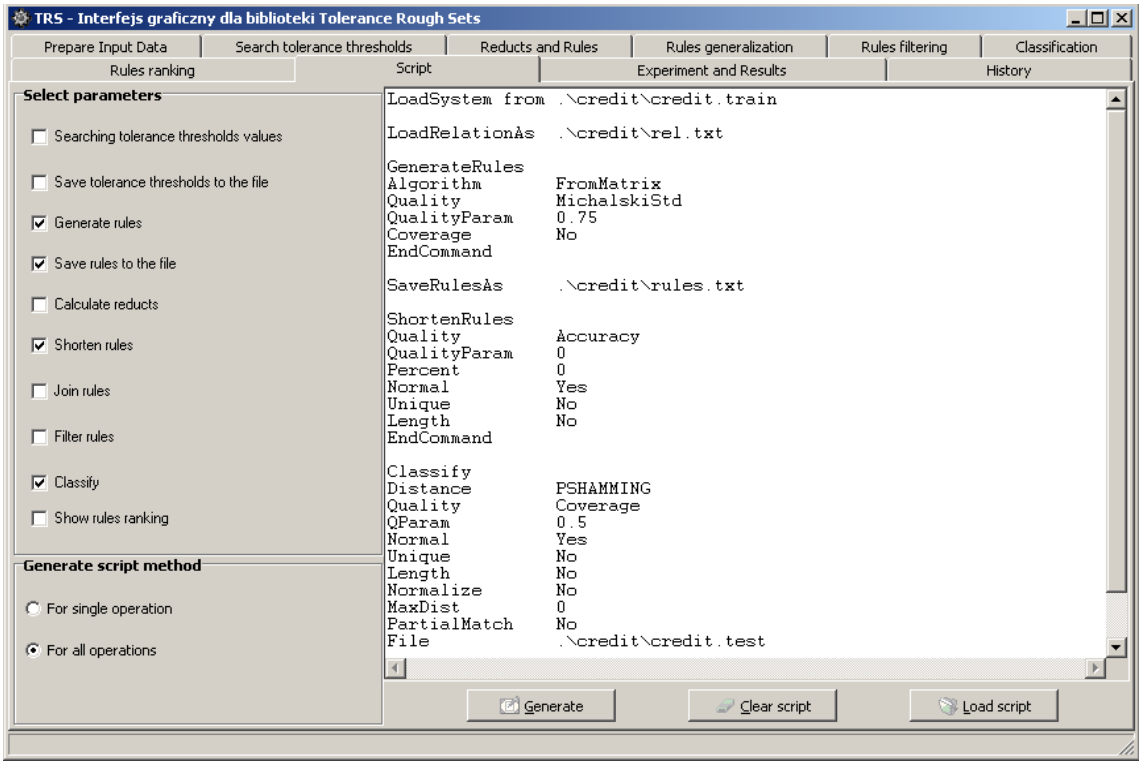

Fig. 2. Generating script tab – it is possible to change script content manually
Rys. 2. Zakładka generowania skryptu – z możliwością ręcznej zmiany zawartości skryptu

After loading data, user gains access to the basic description statistics characterizing the loaded set (min, max, mean, etc) using the context menu.

In every case there is a possibility to load or save analysis results, particularly: tolerance thresholds and determined rules.

All results are saved into text files:

- Out.txt – analysis results: rules, rules after post processing, etc. The order of the file entries is the same as the order of algorithms in the control script. It is possible to open the window (fig. 3) from the "Experiments and Results" tab, what simplifies navigation through the results.

- Ranking.txt – rules and quality measure values according to which ranking was created.
- Rep – results of testing set classification. These results can be also browsed from the "Experiment and Results" tab (fig. 3).
- Rel.txt – tolerance thresholds' values.
- Rules.txt – generated rules.

Rel.txt and Rules.txt files may have text or binary format depending on the user preferences

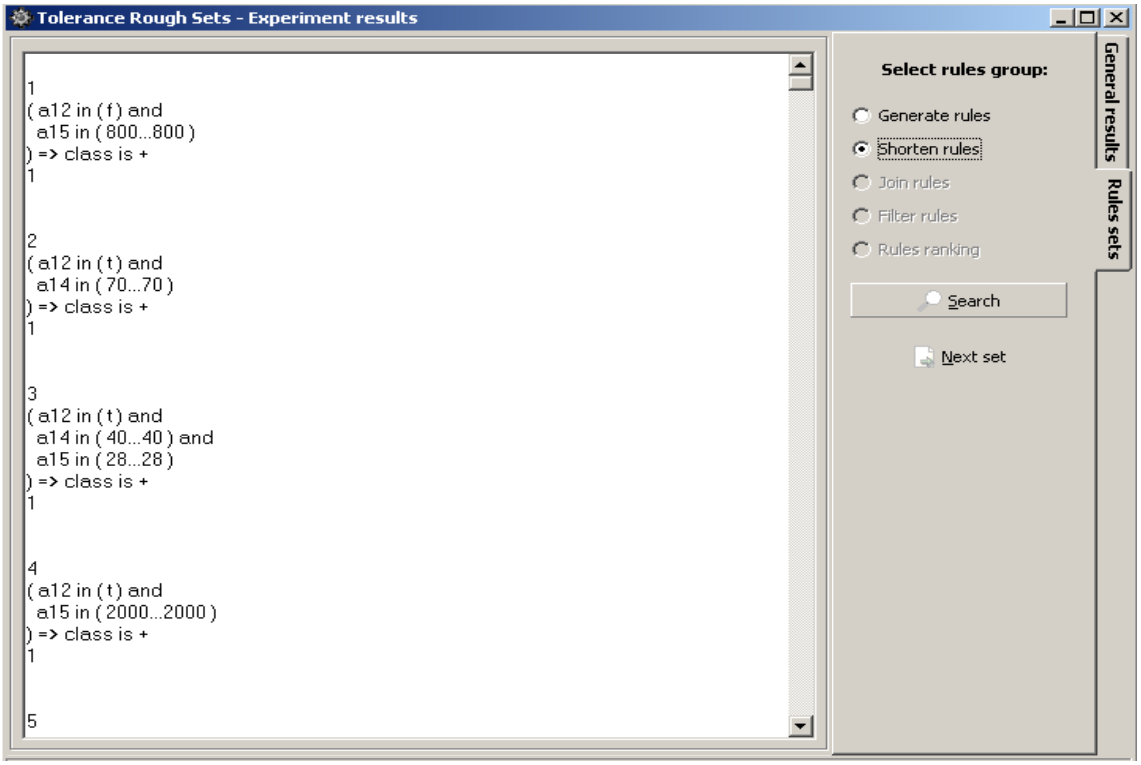

Fig. 3. The form simplifies browsing through analysis results (generated rules sets and summary information – classification accuracy, number of rules, etc.)

Rys. 3. Formatka przeglądania wyników upraszcza analizowanie rezultatów (wygenerowanych zbiorów reguł oraz informacji podsumowujących – dokładność klasyfikacji, ilość reguł etc.)

Results of the experiment are saved into directory specified by the experiment name. Inside the experiment main directory sub-directories: train, test, tune, rel and rules are created. Generated script and script interpreter *trsgo.exe* are also saved into the directory. It allows experienced users to execute experiments without graphical user interface using prepared scripts and data sets.

There is a conf catalog in TRS application directory, which includes among others *Defaults.ini* file containing default values of parameters for individual algorithms.

4. Summary

TRS library described in the paper is the tool for preparing data to induce, analyze and post process decision rules. The library implements some standard algorithms and some new ones proposed by its authors. During presentation of algorithms the literature was mentioned where one can find detailed descriptions of the algorithms and their efficiency. TRS library is accessible in the compiled form and analysis process is defined by controlling scripts. This solution permits using the library by writing direct scripts or by using graphical interface simplifying script generation and interpretation of the results.

Further TRS library development will proceed in two directions. One will consist in adding new algorithms to the library (including rules induction algorithm with continuous decision attribute) and adding mechanisms of distributed algorithms execution over the network. The second direction will be based on improving graphical user interface, as its current version requires some minor modifications. The main modification will insist on adding the possibility of building so-called schemes of analysis to allow defining the process of data analysis as combination of connected icons. Each icon will represent single algorithm and double clicking on the icon will switch user to the corresponding algorithm tab. Then, adding analysis schemes basically will not change the interface, but will eliminate the necessity of switching over different tabs.

The library in its executable form (including graphical user interface) can be obtained from the authors of this article. The authors can also provide help in creating and executing experiments and data analysis. The library co-author is Paweł Proksa, postgraduate student in the Institute of Computer Science.

REFERENCES

1. Bazan J. G., Szczuka M. S., Wróblewski J.: A New Version of Rough Set Exploration System. Lecture Notes in Artificial Intelligence 2475, Berlin, Heidelberg: Springer-Verlag, 2002, pp.397-404..
2. Bruha I.: Quality of Decision Rules: Definitions and Classification Schemes for Multiple Rules. Nakhaeizadeh G., Taylor C. C. (ed.) Machine Learning and Statistics, The Interface. John Wiley and Sons, 1997
3. Grzymała-Busse J.: LERS – a system for learning from examples based on rough sets. Słowiński R. (ed.): Intelligent Decision Support. Dordrecht. Kluwer 1992, pp.3-18.
4. Goldberg D. E.: Algorytmy genetyczne i ich zastosowania. Wydawnictwo Naukowo-Techniczne, Warszawa 1998.

5. Michalski R. S., Bratko I.: Machine Learning and Data Mining: Methods and Applications. John Wiley and Sons, 1998.
6. Nguyen H. S., Nguyen S. H.: Some efficient algorithms for rough set methods. Proceedings of the Sixth International Conference, IPMU'96 2, July 1-5, Granada, Spain pp. 1451-1456.
7. Ohrn A.: Rosetta technical users manual. Knowledge Systems Group, Dept. of Computer and Information Science, NTNU, Norway, 2001. (http://rosetta.sourceforge.net/)
8. Pawlak Z.: Rough sets: Theoretical aspects of reasoning about data. Dordrecht: Kluwer, 1991
9. Proksa P., Sikora M.: Application of Genetic Algorithms to Create Rule Description of Decision Classes. V Krajowa Konferencja Algorytmy Ewolucyjne i Optymalizacja Globalna. Jastrzębia Góra, 30.05-1.06, 2001, pp. 178-196.
10. Sikora M.: Filtracja zbioru reguł decyzyjnych wykorzystująca funkcje oceny jakości reguł. Studia Informatica Vol. 46, No. 4, Gliwice 2001.
11. Sikora M., Proksa P.: Algorithms for generation and filtration of approximate decision rules, using rule-related quality measures. Bulletin of International Rough Set Society Vo. 5, No. 1/2 .Proceedings of RSTGC-2001, 2001, pp.
12. Sikora M., Proksa P.: Induction of decision and association rules for knowledge discovery in industrial databases. DM-IEEE, IEEE International Conference of Data Mining, Brighton, 01-04, November 2004.
13. Sikora M.: Approximate decision rules induction algorithm using rough sets and rule-related quality measures. Archiwum Informatyki Teoretycznej i Stosowanej (w druku).
14. Stefanowski J.: Rough set based rule induction techniques for classification problems. Proc. 6-th European Congress for Intelligent Techniques and Soft Computing, vol.1, Aachen, Sept. 7-10, 1998, pp.107-119.
15. Stepaniuk J.: Knowledge Discovery by Application of Rough Set Models. ICS PAS Reports No. 887, Warszawa, 1999.

Recenzent: Prof. dr hab. Wojciech Moczulski

Wpłynęło do Redakcji 14 wrzesień 2005 r.

Omówienie

W artykule opisano bibliotekę TRS, udostępniającą indukcję reguł decyzyjnych za pomocą tolerancyjnego modelu zbiorów przybliżonych. Zaimplementowano algorytmy,

umożliwiające poszukiwanie optymalnego wektora progów tolerancji, wyznaczania reduktów relatywnych, współczynnika istotności atrybutu, indukcji reguł z tzw. reduktów relatywnych oraz zaproponowany przez autorów algorytm RMatrix.

Ważną częścią biblioteki są zaproponowane przez jej autorów algorytmy, umożliwiające filtrację i uogólnianie wyznaczonych reguł. Ta część biblioteki zawiera algorytmy skracania i sklejania reguł oraz filtracji (z pokrycia, „W przód", „Wstecz"). Wszystkie algorytmy perują na tzw. empirycznych miarach, oceniających jakość reguł, z tego powodu w bibliotece istnieje możliwość obliczania wartości dziesięciu najbardziej popularnych miar.

Ze względu na fakt, iż biblioteka podlega nieustannym modyfikacjom i rozwojowi, obecnie zaimplementowano w niej również algorytmy nie związane bezpośrednio z tolerancyjnym modelem zbiorów przybliżonych. Przykładem takiego algorytmu może być algorytm MODLEM, który wyposażono w dodatkowe parametry.

Dotychczas korzystanie z biblioteki było możliwe za pośrednictwem wersji skompilowanej lub poprzez opracowany interpreter skryptów. Obecnie opracowano pierwszą wersję umożliwiającą korzystanie z biblioteki za pośrednictwem interfejsu graficznego, dzięki czemu możliwe jest łatwiejsze przeprowadzanie eksperymentów i analiz. Dokładniej, opracowano program umożliwiający przygotowanie danych oraz generujący w sposób automatyczny skrypty sterujące przebiegiem analizy. Dalsze prace nad rozwojem biblioteki przebiegać będą dwutorowo. Pierwszy kierunek – to wyposażanie biblioteki w coraz to nowe algorytmy (w tym algorytm indukcji reguł z ciągłym atrybutem decyzyjnym). Drugi – to znacznie udoskonalenie interfejsu użytkownika (w tym o możliwość budowania tzw. schematów analizy).

Adresy

Marek SIKORA: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16,
44-100 Gliwice, Polska, Marek.Sikora@polsl.pl .
Aleksandra GRUCA: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16,
44-100 Gliwice, Polska, Aleksandra.Gruca@polsl.pl .
Robert GRUCA: Mentor Graphics Polska Sp. z o.o., ul. Chorzowska 50,
40-121 Katowice, Polska, Robert_Gruca@mentor.com