Łukasz NOZDRZYKOWSKI, Magdalena NOZDRZYKOWSKA
MARITIME UNIVERSITY OF SZCZECIN, INSTITUTE OF MARINE TECHNOLOGIES
1-2 Wały Chrobrego St., 70-500 Szczecin, Poland

Implementation of the attribute significance analysis with the use of soft reduction of attributes in the rough set theory on the basis of the SQL mechanisms

Abstract

This article presents a way to use databases supporting the SQL and PL/SQL in the implementation of a method of attribute significance analysis with the use of soft reduction of attributes in the rough set theory. A number of SQL queries are presented, which facilitate the implementation. The original mechanisms presented previously [1] are supplemented with queries which facilitate the execution of attribute coding. The authors present a complete implementation of the method, from the coding of attributes to the determination of the significance of conditional attributes. Application of queries to the database eliminates the necessity to build data grouping and data mining mechanisms and calculation of repetitions of identical rules in the reduced decision rule space. Without the support of a database, the creation of universal data grouping and data mining mechanisms which could be used with any number of attributes is a challenging task.

Keywords: rough set theory, data analysis, soft reduction of attributes, SQL implementation.

1. Introduction

Data analysis methods enable us to obtain useful information from the collected data. Depending on the type of data, various data analysis methods are applied [2], e.g. statistical, exploratory, and other. Statistical methods are used for describing mass phenomena and their changes in a predefined time limit. Exploratory methods, based on data mining with the use of software tools, detect patterns in the collected data, hidden to the human mind. Data exploration uses various techniques and methods, such as visualization, statistical methods, neural networks, machine learning methods, evolutionary methods, fuzzy logic and rough sets [3].

The rough set theory, proposed by Prof. Zdzisław Pawlak, is mainly used for the synthesis of effective analysis methods and reduction of data sets. By means of mathematical tools, such as quick and easy data analysis algorithms, it enables us to obtain meaningful knowledge from the collected data. It identifies partial and full dependencies, facilitates the determination of non-numeric values, missing data and dynamic data [4].

The article presents the application of relational databases supporting the Structured Query Language (SQL) in full software implementation of the analysis of significance of conditional attributes in the rough set theory, on the basis of soft reduction of attributes in rough sets. Implementation of the analysis can be significantly simplified owing to the capabilities of the SQL. The article presents the results of continued work on the implementation [1]. Mechanisms facilitating data discretisation and coding have been added. The proposed solution is universal in character and reduces the time and costs of implementation of individual stages of the algorithm. Owing to its universal character, the proposed method of analysis can be used in various scientific disciplines.

2. Significance analysis with the use of soft reduction of attributes in the rough set theory

The first step in data mining is the creation of an info array of examples with conditional attributes and the class label attribute. The info array should be in the form of a set expressed by the formula $T= (U,Q,D,V,f)$, where: U – a set including all examples, Q – a set of attributes, D – a set of class label attributes, V – a set of all possible attribute values, f – the info function [5].

Both conditional and class label attributes may be presented as numeric or linguistic values. Conditional attributes are divided into sub-sets whose number depends on the diversification of data. Data is divided with the use of one of the data discretisation methods presented in chapter 4. Each sub-set is assigned a coded value, whilst decoded conditional attributes and class label attributes are written down in a separate array. In step two, elementary conditional sets are determined, i.e. sets containing identical coded conditional attributes, and designated with Ei, where i is the consecutive set number. Additionally, a conditional info array $P=\{q_1, q_2, q_3\}$ is created in the space, which shows the number of examples in each elementary conditional set. The same is done to class label attributes – elementary class label sets are created and class label concept families are determined and designated with Xi, where i is the concept number. The reduced sets are used for the formulation of certain rules, which serve as a basis for the determination of significance of reduced attributes in terms of retaining the quality of the generated rules.

In step three, the significance of attributes is determined through soft reduction of conditional attributes based on relative probability in the rough set theory. This is a way to determine which attributes may be rejected without causing a decrease in the number of examples generating certain rules. An attribute may be reduced also in a situation where the decrease in the number of examples generating certain rules is acceptable. The quality of rules is assessed by determining the relative probability of atomic rules, expressed with the following formula:

$$P_w = \frac{P}{L} \tag{1}$$

where:

P – the total of probabilities of atomic useful rules,
L – the number of elementary conditional sets.

A rule is considered useful if the probability is higher than a predefined value. Attempts of reduction of the number of attributes are made by eliminating one parameter after another and determining their relative significance [6].

3. Application of the SQL in the implementation of the method of soft reduction of attributes in the rough set theory

The method of attribute significance analysis with the use of soft reduction of attributes in the rough set theory, although using simple algorithms for the transformations, poses a challenge for the programmer due to its advanced computations and the required memory capacity. The duration of computations increases significantly in line with the rising number of attributes subject to reduction. The programmer must also secure the storage capacity for large amounts of data corresponding to individual rules. Additionally, implementation of the method for various numbers of attributes is impeded by the nature of the method, which requires the use of program loops nested within one another in a number corresponding to the number of attributes under examination. Increasing the number of attributes by one requires

adding another loop and restructuring the data arrays and the auxiliary functions. Taking all this into consideration, it is difficult to write a program code which would be universal. Another problem is posed by the auxiliary algorithms required for the implementation of the analysis method. It is necessary to create data grouping mechanisms and data mining mechanisms which will sum up the repeating instances of identical rules in the reduced space of decision rules. These operations, although simple to define and carry out by a human being, are difficult to implement, especially in a universal manner for any number of attributes and any number of rules.

The authors propose a way to simplify the process of implementation of the attribute significance analysis algorithm with the use of soft reduction of conditional attributes in the rough set theory by leveraging the mechanisms available in relational databases and the capabilities of the SQL. The implementation methods presented at the Transcomp 2016 conference [1] have been extended with methods of discretisation of variables and coding which leverage the capabilities of databases, to facilitate full implementation of the presented method in any high-level programming language applied in SQL-based databases. Such a program can facilitate the significance analysis under certain additional conditions. The mechanisms which execute SQL queries in the database must support the execution of conditional CASE expressions and the definition of input/output functions (PL/SQL or similar implementations). Functions defined on the database side can be applied in the program and serve as tools for data mining. Databases such as PostgreSQL [7] and Oracle Database support function-defining features. It should be noted here that some databases may implement function-defining features using a language other than the PL/SQL from Oracle. The authors utilized PostgreSQL 9.5.4 and its function-defining features. Other database systems will require the development of appropriate functions on the basis of the presented code. Additionally, the C++ language, available in the Microsoft Visual Studio 2015 programming environment, has been used to verify the implementation and carry out the efficiency tests.

4. Discretisation of variables for the analysis

The first step in the attribute significance analysis with the use of the rough set theory is discretisation of continuous variables, which are necessary for coding conditional and class label attributes. Discretisation can be performed for an odd or even number of classes by means of several methods: by dividing the variables into equal intervals, by the method of equal number of samples in each class, or, additionally for an even number of classes, by the median value method. The two former methods are implemented in the same way for an odd and even number of classes. As a result of discretisation, the Base Table is created with coded conditional and class label attributes, to be used in further analysis. In this chapter, methods supporting discretisation of variables are presented. Further in the article, it has been assumed that non-coded data is stored in the inputData table, whilst coded data – in the Base Table. Data in the inputData table is sorted in columns (col_1, col_2,…col_n) by attributes. Data sets may be divided through expert evaluation or by one of the methods of discretisation of continuous variables.

The simplest discretisation can be carried out by dividing the continuous variables into intervals. Interval boundaries are determined through equal division of the set value, regardless of the number of samples in individual intervals. The method is expressed by the following formula (2):

$$d = (Xmax - Xmin)/k \tag{2}$$

where:

d – interval width, $Xmin$ – minimum value
$Xmax$ – maximum value, k – number of intervals.

A downside to this method is that some of the intervals obtained may have no examples or a small number of them [8]. The first step in this type of discretisation consists in the determination of the interval width for individual attributes. For this purpose, the creation of a function returning the interval width is proposed, as shown in Listing 1.

Listing 1. Function determining the interval width

```
    CREATE FUNCTION width1(integer) RETURNS setof double
precision AS
    $$
    SELECT MAX(col1)/$1 FROM inputData
    $$
    LANGUAGE SQL;
```

This listing presents the function for the first column assigned to the first attribute subject to discretisation. The input parameter here is the number of intervals into which the set will be divided. This number is used to determine intervals corresponding to subsequent boundary values. The boundary values are multiples of the interval width. For the purpose of determining the boundary values, another function is proposed, shown in Listing 2 for the first attribute.

Listing 2. Function determining interval boundaries

```
    CREATE FUNCTION range1(integer, integer) RETURNS setof tab
AS
    $$
    SELECT min(col1), max(col1) FROM inputData WHERE
    col1 > $1*(SELECT * FROM width1 ($2))-(SELECT * FROM width1
($2)) AND
    col1 <($1+1)*(SELECT * FROM width1 ($2));
    $$
    LANGUAGE SQL;
```

The input parameters for this function are the interval number and the number of classes into which the set is divided. An example of the function call for the second of three intervals determined is shown in Listing 3.

Listing 3. Interval boundary determining function call

```
    SELECT * FROM range1 (2,3);
```

Discretisation by means of the method of equal number of samples in classes is more difficult to implement, even more so since dividing a set into perfectly equal intervals is not always possible. By this method, boundaries between individual intervals are supposed to divide the set into intervals of equal number of samples. For a set with n samples, where the number of intervals equals k classes, discretisation by the method of equal number of samples will result in a division where each interval (class) contains n/k samples. Additionally, an accuracy threshold at which the division of sets is satisfactory should be determined.

To implement this method, it is necessary to define an auxiliary function for the calculation of the desired number of samples in each interval. The function is presented in Listing 4.

Listing 4. Function determining the number of samples in intervals

```
    CREATE FUNCTION numberOfSamples(integer) RETURNS setof
bigint AS
    $$
    SELECT COUNT(col1)/$1 FROM inputData
    $$
    LANGUAGE SQL;
```

The parameter of this function is the number of intervals into which the input data will be coded. The output is the approximate number of samples resulting from the discretisation of a given attribute.

Another auxiliary function, presented in Listing 5, is the function determining the actual number of samples in a given interval.

Listing 5. Function determining the number of samples in intervals

```
    CREATE FUNCTION samplesInRange(real, real) RETURNS setof
bigint AS
    $$
    SELECT COUNT(col1) FROM inputData WHERE col1>$1 AND
col1<=$2
    $$
    LANGUAGE SQL;
```

Its input parameters are the lower and upper limit; the output is the number of samples in the thus determined interval. The desired width and boundaries of the interval can be determined with the use of this function and a while loop. The examined interval range is gradually increased in the while loop until the desired interval width is obtained.

The last method of discretisation presented here is the median value. For this method, median values from individual intervals must be determined by recursion. Discretisation by the median value method consists in the determination of sample limits on the basis of the calculated median value. The median value divides a set into two sub-sets, which can then be divided again with the use of the median value in a given sub-set. By this method, a set can be divided into an even number of sub-sets only. The median value is determined with the following formula (3):

$$E(X) = \sum_{i=1}^{n} x_i p_i \tag{3}$$

where:

n – the number of values of the random variable X,
x_i – the i^{th} value of the random variable X, $i = 1, 2, \ldots, n$
p_i – the probability of the value of x_i

This method in fact calculates the median value, since probabilities for the samples are equal. The median value in a given interval can be calculated with the function shown in Listing 6.

Listing 6. Function determining the interval median value

```
    CREATE FUNCTION average(real, real) RETURNS setof bigint AS
    $$
    SELECT AVG(col1) FROM inputData WHERE col1>=$1 AND col1<$2
    $$
    LANGUAGE SQL;
```

The parameters of this function are the lower and upper boundaries of the interval. The median value can be used to determine boundaries for individual classes.

The discretisation methods presented above enable us to determine intervals into which all the attributes will be coded. For this purpose, SQL INSERT queries need to be created. The data should be recorded in a new Base Table, developed as shown in Listing 7.

Listing 7. Creation of a Base Table

```
    CREATE TABLE Base
     {k0 integer DEFAULT 0, k1 integer DEFAULT 0,…, kn integer
DEFAULT 0}
```

In the Base Table, the class label attribute is coded first, designated with $k0$, followed by conditional attributes $k1 – kn$. As it is shown in Listing 1, the Base Table contains data coded as integers.

For easier implementation, the authors propose the storage of coded attributes as a collection of natural numbers from 1 to k, where k is the number of consecutive encoding levels. It will facilitate the execution of program loops performing the analysis.

5. Determination of relative probability of all atomic useful rules

The Base Table should be reduced to a simplified form by deleting the repeating rules. The new table should include only unique rules with the number of repetitions for each rule in a separate column. Rules are understood as table rows which show how coded attributes translate into the coded class label attribute. The structure of the new table, called Elementary Table, is presented in Listing 8.

Listing 8. Creation of an Elementary Table

```
    CREATE TABLE ElementaryTable
     { distinctAttr  integer DEFAULT 0,
    countAttr integer DEFAULT 0,
    k0 integer DEFAULT 0, k1 integer DEFAULT 0,…, kn integer
DEFAULT 0}
```

The Elementary Table stores data with coded conditional attributes and the class label attribute as well as the number of repetitions of individual rules included in the original Base Table. The new table has the following structure:

– distinctAttr – coded class label attribute,
– countAttr – number of individual unique rules,
– $k1$- kn – coded conditional attributes.

In order to determine unique rules, a SELECT query must be executed from the Base Table, with the use of grouping over all attributes. On the basis of the output data, INSERT orders must be created for the new table. The SELECT query is shown in Listing 9.

Listing 9. A query generating data into an ElementaryTable

```
    SELECT DISTINCT(k0), COUNT (k0), k1, k2…, kn FROM Base
GROUP BY k0, k1, … , kn;
```

The next step is determination of basic analysis parameters for the generated data, with the use of rough sets. The following data must be generated:

– number of elementary sets,
– number of useful rules,
– total absolute probability of atomic useful rules,
– relative probability of all atomic useful rules.

At this stage, the presented parameters must be determined on the basis of all the possible combinations of values for the coded conditional attributes. It can be done by using program loops nested within one another, in a number equal to the number of conditional attributes. Each program loop should iterate over all code values for individual attributes. A diagram is shown in Listing 10, where, as an example, attributes are coded with values from 1 to 5.

Listing 10. Example of program loops generating queries with all combinations of attributes

```
    for (indexK1=1; indexK1<5; indexK1++)
          for (indexK2=1; indexK2<5; indexK2++)
                …
               for (indexKn=1; indexKn<5; indexKn++)
    {
                    //loop body

    }
```

The body of the loop at the deepest level of nesting is responsible for the determination of the values of individual analysis parameters. The respective query is shown in Listing 11.

Listing 11. Query to a database counting all of the parameters

```
    SELECT distinctAttr, countAttr FROM ElementaryTable WHERE
k1=indexK1 AND k2=indexK2 AND... AND kn=indexKn
```

This query enables us to determine the number of elementary sets. Each result returned by the query from Listing 10 means that there is a rule existing in the database. The number of elementary sets is then increased by one.

The number of useful rules is determined on the basis of certain rules, taking into consideration uncertain rules of high probability of occurrence. A certain rule is a rule from a well-defined part, without contradiction. An uncertain rule is useful if it comes from a not well-defined part and generates contradictions as to the class label attribute, but its probability of occurrence for a given value of the class label attribute is below a predefined threshold. A rule is considered useful if, for a certain number K of examples of the rule defined by conditional and class label attributes and for a certain number L of examples of the rule defined only by conditional attributes (the number of elementary sets), the K/L ratio is higher than an expert-predetermined threshold T.

The parameter L is obtained through the query shown in Listing 12.

Listing 12. A query calculating the parameter L

```
    SELECT SUM(countAttr) FROM ElementaryTable WHERE k1=indexK1
AND k2=indexK2 AND... AND kn=indexKn
```

The value of the parameter K is generated through the execution of an SQL query with an additional conditional expression. The query is shown in Listing 13.

Listing 13. Query calculating parameter K

```
    SELECT
          COUNT(
                    CASE WHEN
    ((countAttr::float)/L)>T
    THEN
    ((countAttr::float)/L)
     END
          )    FROM    ElementaryTable    WHERE    k1=indexK1    AND
k2=indexK2 AND... AND kn=indexKn
```

Through a modification of the query, the value of the total absolute probability of useful atomic rules can be obtained. The code in Listing 14 presents a query for partial total probability for the coded values of attributes selected for the query by program loops. The final value of the probability equals the sum of partial values from all the queries.

Listing 14. A query calculating the total absolute probability of useful atomic rules

```
    SELECT
          SUM(
                    CASE WHEN
     ((countAttr::float)/L)>T
    THEN
    ((countAttr::float)/L)
    END
          )    FROM    ElementaryTable    WHERE    k1=indexK1    AND
k2=indexK2 AND... AND kn=indexKn
```

Both queries from Listings 13 and 14 can be completed with the query from Listing 12, thus reducing the total number of queries in the program. Listing 15 presents a modified query from Listing 13.

Listing 15. A query generating full absolute probability of atomic rules

```
    SELECT
          SUM(
                    CASE WHEN
    ((countAttr::float)/
    SELECT SUM(countAttr) FROM ElementaryTable WHERE k1=indexK1
AND k2=indexK2 AND... AND kn=indexKn
    ) > T
    THEN
    ((countAttr::float)/
    SELECT SUM(countAttr) FROM ElementaryTable WHERE k1=indexK1
AND k2=indexK2 AND... AND kn=indexKn
    )
    END   )    FROM    ElementaryTable    WHERE    k1=indexK1    AND
k2=indexK2 AND... AND kn=indexKn
```

No additional queries are required to generate the relative probability of useful rules, since it is determined as the ratio of the full absolute probability of atomic rules to the number of elementary sets.

6. Soft reduction of conditional attributes in the rough set theory

By the method using soft reduction of conditional attributes in the rough set theory, consecutive conditional attributes are rejected in order to verify whether the reduction does not generate a significant decrease in the relative probability of atomic useful rules. A set is reduced by an attribute provided that the resulting decrease in the relative probability is insignificant; otherwise, the attribute is not deleted. The rejection threshold may be defined through expert attempts.

In order to carry out reduction attempts, the authors propose to create a new table to be filled in with data on rules in the reduced space of attributes. The assumption is that the table is filled in with data from the Base Table, and the selected attribute subject to reduction is skipped. After a reduction attempt, the table should be cleaned before a new attempt of reduction of the next conditional attribute is performed. The structure of the table is similar to that of the Elementary Table, except for the number of fields for the coded conditional attributes, which is reduced by one. The SQL code creating the Reduction Table is shown in Listing 16.

Listing 16. A query forming a table after an attempt to reduce a selected attribute

```
    CREATE TABLE ReductionTable
          { distinctAttr  integer DEFAULT 0,
    countAttr integer DEFAULT 0,
    k0 integer DEFAULT 0, k1 integer DEFAULT 0,…, kn-1 integer
DEFAULT 0}
```

The input data for the Reduction Table is generated through a SELECT query to the Base Table. The query is shown in Listing 17.

Listing 17. A query obtaining data for the reduction table

```
    SELECT  DISTINCT(k0),  COUNT  (k0),  k1,  k2…,  kn  FROM  Base
GROUP BY k0, k1, … , kn;
```

Now, it is necessary to determine all the parameters again, i.e. the number of elementary sets, the number of useful atomic rules, the total probability of useful rules, and the relative probability of atomic useful rules. Additionally, the significance of the thus reduced attribute is determined, to evaluate whether it can be reduced.

Similarly to the analysis with the full set of attributes, the analysis with the reduced number of attributes uses loops nested within one another, in a number n-1, corresponding to the reduced (by one) number of conditional attributes in the set. The further analysis uses the same queries which were run in the full space of conditional attributes. The only difference lies in the execution of queries to the Reduction Table.

The dependence shown in formula 4 is applied to determine the significance of the reduced attribute.

$$\text{significance}(q_i) = \frac{p_{\text{full}} - p_{\text{red}_{qi}}}{p_{\text{full}}} \quad (4)$$

where:

- p_{full} - relative probability of atomic useful rules for a full set
- $p_{\text{red}qi}$ - relative probability of atomic useful rules for a set reduced by the i^{th} conditional attribute q_i.

Reduction of an attribute is considered possible if the attribute significance is low. However, considering the possibility of a material decrease in the total relative probability of rules, it is important that before a decision is made to reduce a set by more than one attribute of low significance, an attempt to reduce the set by more than one attribute at a time is made with the use of the formula presented above.

7. Summary

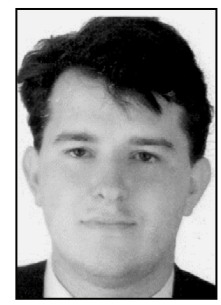
The article presents a solution facilitating the implementation of the method of determining the significance of conditional attributes through their soft reduction in the rough set theory. For this purpose, mechanisms available in the SQL and relational databases have been used. The implementation is made easier owing to the utilization of advanced mechanisms of data mining, including grouping and aggregation functions. The necessary competence of the programmer is limited to simple steering operations in the selected programming language and understanding the execution of queries to the selected database.

8. References

[1] Nozdrzykowski Ł, Wróbel M.: The use of SQL as a tool supporting the implementation of a method of attribute significance analysis based on soft re-duction of attributes in the rough set theory, TTS 12/2016.
[2] Makowska M: Analiza danych zastanych. Przewodnik dla studentów. Wydawnictwo Naukowe Scholar, 2003.
[3] Hand D., Mannila H., Smyth P.: Principles of Data Mining, MIT Press, 2001.
[4] Pawlak Z.: Zbiory przybliżone nowa matematyczna metoda analizy danych. In: Miesięcznik Politechniki Warszawskiej, no 5/2004. http://bcpw.bg.pw.edu.pl/Content/1949/zb_przyb.pdf.
[5] Łatuszyńska M., Wawrzyniak A., Wąsikowsk B, Galindo E., Sandoval J.: Teoria zbiorów przybliżonych w wykrywaniu reguł zachowań zakupowych kobiet i mężczyzn podczas kupowania telefonów komórkowych. Zeszyty Naukowe Uniwersytetu Szczecińskiego, Studia Informatica No 35, 2014.
[6] Piegat A.: Matematyka-zbiory przybliżone. Materiały do wykładów. Politechnika Szczecińska, Wydział Informatyki, Szczecin, 2007.
[7] Żmuda K.: SQL, Jak osiągnąć mistrzostwo w konstruowaniu zapytań, Helion 2015.
[8] Kaya F.: Discretizing continuous features for naïve Bayes and C4. 5 classifiers. University of Maryland publications, College Park, 2008.

Received: 02.10.2016 *Paper reviewed* *Accepted: 02.12.2016*

Łukasz NOZDRZYKOWSKI, PhD, eng.



He graduated in computer science from West Pomeranian University of Technology, obtaining MSc degree in 2006 and PhD degree in 2013. He is currently a Deputy Director at the Institute of Marine Technologies in Maritime University of Szczecin. His research interest include cryptography, steganography and parallel computing.

e-mail: l.nozdrzykowski@am.szczecin.pl

Magdalena NOZDRZYKOWSKA, MSc, eng.

Magdalena Nozdrzykowska has graduated in computer science from West Pomeranian University of Technology in 2011. She is currently assistant at the Institute of Marine Technologies in Maritime University of Szczecin. Her research interest parallel computing and parallel programming.

e-mail: m.nozdrzykowska@am.szczecin.pl