AN ALGORITHM FOR EXTRACTION OF RULE SETS DIRECTLY FROM NUMERICAL DATA SET BASED ON ROUGH SET THEORY

P. Kaločay[1], *M. Turčaník*[2]

[1]Department of Technical Support Systems and Automatization,
[2]Department of Informatics and Computer Sciences,
Military academy, Liptovský Mikuláš, Slovak republic

ABSTRACT: *In this paper there is introduced an algorithm for extracting two kinds of rule sets directly from numerical data set based on rough set theory. This algorithm stands for recursive process in which remaining set of objects belonging to a particular class is either approximated if in that set there are some objects belonging to another class or not. In the first case a rule is defined by the lower approximation of that set otherwise a rule is defined by the crisp set. This way a minimal rule set is extracted from the whole set of objects. For characteristic rule set only rules with higher strength than specified one are extracted. The algorithm performance was verified on the Iris data set. This algorithm is suitable for solving classification tasks.*
Keywords: *rough set theory, rule extraction, classification*

1. Introduction

Solving classification tasks occurs in wide range of human activity such as decision making processes, diagnosis etc. Various means can be used but one have to take into account their pros and cons for the concrete real-life problem to be solved. One of suitable means is rough set theory (RST), which can be used even under condition of imprecise and uncertain data.

The original rough set philosophy was founded on the assumption that objects characterized by the same values of attributes are indiscernible and therefore the whole set of objects can be divided into elementary sets. This division is either exact if a particular elementary set contains only objects assigned to the same class (crisp set) or approximate otherwise (rough set). Any rough set can be replaced by a pair of crisp sets, called lower and upper approximation. The lower approximation consists of all objects which surely belong to the set and the upper approximation contains objects which possibly belong to the set. A difference between upper and lower approximation is called boundary region and for rough set this region is not empty. An exact rule can be defined for each lower approximation and an approximate rule for upper approximation. Each rule can be characterized by strength, defined as the number of objects covered by this rule [2, 4, 5, 7].

However the original concept of this theory is not suitable in the cases when there is a relatively large set of objects because of imprecise measurements, random fluctuation of attribute values etc. This fact results in huge number of rules [4, 5]. Therefore a new approach was introduced into RST which is based on similarity and reflexivity relation to define rough approximation of sets of objects with similar or the same attribute values. For example the goal of the algorithm called Minimal description-heuristic is to found out a minimal set of cuts on attribute value intervals. By these cuts the whole set of objects is divided into minimal number of elementary sets. Finally a rule can be defined for each elementary set [4]. In another algorithm tolerance intervals are determined for attribute values. From these tolerance intervals rough approximations of sets of objects with similar or the same attribute values are

created. In addition three kinds of exact rule sets can be induced – a minimal, an exhaustive and a characteristic rule set [5].

In this article there is described an algorithm for extracting two kinds of exact rule sets from numerical data set – a minimal rule set with minimal number of rules and a characteristic rule set with only such rules of which the strength is higher than that one specified by user.

Following section of this article is devoted to the description of this algorithm. Then in section 3 the classification ability of the algorithm is verified on the Iris data set. Pros and cons of using this algorithm are sumarized in section 4.

2. The algorithm description

The idea of creating this algorithm came from [1] where fuzzy classifier with activation and inhibition hyperboxes is described. An algorithm of that classifier is able to generate those hyperboxes, create membership functions for them and tune their parameters. A set of fuzzy rules is the final product. But in this approach a rule set can be generated in a simplier way.

Partitioning set of objects is discussed first. Let´s have at disposal a numerical data set X in which objects are represented by examples. Each example is described by m-dimensional vector of attribute values $\boldsymbol{x} = \{x_1, x_1, \ldots, x_m\}$ and assigned to one of n classes. Let X_i be a set of examples assigned to class i and let $A_i(1)$ be the m-dimensional subspace of level 1 determined by minimal and maximal attribute values of X_i.

$$A_i(1) = \{\boldsymbol{x} \mid v_{ik}(1) \le x_k \le V_{ik}(1),\ k = 1, \ldots, m\} \tag{1}$$

where: x_k = kth element of vector $\boldsymbol{x}$ ($k = 1, \ldots, m$),
$v_{ik}(1)$ = minimal value of x_k of $\boldsymbol{x} \in X_i$,
$V_{ik}(1)$ = maximal value of x_k of $\boldsymbol{x} \in X_i$.

If in this subspace there is not any example assigned to class j ($j \neq i$, $j = 1, \ldots, n$), then the set X_i is the crisp set and a rule of level 1 for class i is defined as follows:

$$\text{if } \boldsymbol{x} \text{ is in } A_i(1) \text{ then } \boldsymbol{x} \text{ belongs to class } i \tag{2}$$

If in the subspace $A_i(1)$ there is an example assigned to class j ($j \neq i$, $j = 1, \ldots, n$), then the set X_i has to be approximated. The defined subspace $A_i(1)$ is then equal to the upper approximation of X_i marked $A_i^*(1)$ and a boundary region has to be determined. This boundary region is denoted $Bn_{ij}(1)$:

$$Bn_{ij}(1) = \{\boldsymbol{x} \mid u_{ijk}(1) \le x_k \le U_{ijk}(1),\ k = 1, \ldots, m\} \tag{3}$$

where $v_{ik}(1) \le u_{ijk}(1) \le U_{ijk}(1) \le V_{ik}(1)$. Minimal and maximal attribute values of boundary region are determined as follows:

1. For $v_{jk}(1) \le v_{ik}(1) \le V_{jk}(1) < V_{ik}(1)$

$$\begin{aligned} u_{ijk}(1) &= v_{ik}(1) \\ U_{ijk}(1) &= V_{jk}(1) + \alpha[V_{ik}(1) - V_{jk}(1)] \end{aligned} \tag{4}$$

2. For $v_{ik}(1) < v_{jk}(1) \le V_{ik}(1) < V_{jk}(1)$

$$\begin{aligned} u_{ijk}(1) &= v_{jk}(1) - \alpha[v_{jk}(1) - v_{ik}(1)] \\ U_{ijk}(1) &= V_{ik}(1) \end{aligned} \tag{5}$$

3. For $v_{jk}(1) \le v_{ik}(1) \le V_{ik}(1) < V_{jk}(1)$

$$\begin{aligned} u_{ijk}(1) &= v_{ik}(1) \\ U_{ijk}(1) &= V_{ik}(1) \end{aligned} \tag{6}$$

4. For $v_{ik}(1) < v_{jk}(1) \le V_{jk}(1) < V_{ik}(1)$

$$\begin{aligned} u_{ijk}(1) &= v_{jk}(1) - \alpha[v_{jk}(1) - v_{ik}(1)] \\ U_{ijk}(1) &= V_{jk}(1) + \alpha[V_{ik}(1) - V_{jk}(1)] \end{aligned} \tag{7}$$

where α ($0 < \alpha < 1$) is an user specified parameter for determining the size of $Bn_{ij}(1)$.

For rough approximation is then a rule of level 1 defined in form:

$$\text{if } \boldsymbol{x} \text{ is in } A_i^*(1) \text{ and } \boldsymbol{x} \text{ is not in } Bn_{ij}(1) \text{ then } \boldsymbol{x} \text{ belongs to class } i \qquad (8)$$

i.e. an example $\boldsymbol{x}$ is in a lower approximation of X_i.

If (6) holds for all k ($k = 1, ..., m$), then $A_i^*(1)$ is equal to $Bn_{ij}(1)$. In this case no rule is defined, since no $\boldsymbol{x}$ can be in lower approximation.

If some examples belonging to X_i exist in $Bn_{ij}(1)$ then the m-dimensional subspace of level 2 denoted $A_i(2)$ is determined from those examples. The same procedure is performed until a boundary region of certain level l is empty or any rule can be generated.

Now the structure of the algorithm written in the form of pseudocode follows:

```
begin
  for i:=1 to n do
  begin specify set of objects X_i for class i;
      for j:=1 to n do
      begin if j≠i then do
          begin specify set of objects X_j for class j;
              X_i':= X_i; level:=0;
              while X_i' is not empty
              begin level:=level+1;
                  determine A_i(level) and A_j(level);
                  if A_i(level) covers no example from X_j then define rule (1);
                  if A_i(level) covers any example from X_j then do
                  begin determine A_i^*( level) and Bn_ij(level);
                      if no rule can be generated then break;
                      else define rule (2);
                  end
                  remove covered examples by rule (1) or (2) from X_i;
              end
          end
      end
      for r:=1 to number_of_generated_rules do
      begin count correctly and incorrectly classified examples by rule r;
          while X_i is not empty do
          begin select an exact rule with highest strength;
              remove examples covered by this rule;
              if there is no exact rule then do
              begin select an approximate rule with the smallest number of
                      misclassified examples;
                  determine a class j (j ≠ i, j = 1, ..., n) of incorrectly classified
                      examples; determine level;
                  increase corresponding boundary region Bn_ij(level) until that rule
                      becomes an exact one;
                  if no rule can become an exact one then break;
              end
          end
      end
  end
end
```

The structure of the algorithm can be divided into two parts. In the first part all rules are generated by the process of partitioning set X into X_i and X_j ($j \neq i$, $j = 1, ..., n$). Then in the second part exact rules are selected to insert them into rule set. If there is no exact rule for remaining set of examples X_i, then boundary regions corresponding to approximate rules have to be increased in order to these rules become exact ones.

If a characteristic rule set was requested to extract, then the algorithm does not extract rules with smallest strength than that one specified by user.

3. Verifying the classification ability of presented algorithm

To verify the classification ability of the presented algorithm an experiment was carried out on one of the most used data set Iris [6]. The characteristics of this data set are: 150 objects, 4 attributes a 3 classes.

To fulfil the requirement for creating of learning and testing sets, the cross validation method was used [3]. The whole data set was split into ten subsets and ten experiments were carried out in which 9 of 10 subsets were used as learning set a remaining subset for testing set (10 fold CV).

In table 1 there are results for different values of parameters α. In the second and third column there are average numbers of correctly classified objects and adequate values in percents for learning and testing set respectively. Average numbers of misclassified objects for the testing sets are in the fourth column. Average numbers of not classified objects for the testing sets are in the fifth column. In the sixth column there are average numbers of extracted rules and in the last column there are average values of time in seconds.

Value of parameter α	Learning set	Testing set	Number of misclassified objects	Number of not classified objects	Number of rules	Time [sec]
0,1	134,2 (99,41%)	12,8 (85,33%)	0,9	1,3	8,1	0,2
0,3	130,3 (96,52%)	12,9 (86,0%)	0,4	1,7	7,1	0,2
0,5	134,3 (99,48%)	12,3 (82,0%)	0,6	2,2	9,6	0,24
0,7	135 (100%)	11,9 (79,33%)	0,4	2,8	11,8	0,24
0,9	135 (100%)	10,7 (71,33%)	0,7	3,6	16	0,32

Table 1 – Results for 10 fold CV and different values of parameter α

The results from the table for increasing the value of parameter α can be generalized:

- the number of extracted rules increasing. This is due to increasing boundary regions at the expense of lower approximation of each set.
- the classification accuracy on testing set decreasing because rules cover less m-dimensional space by lower approximations.
- time is increasing too.

Algorithm programming and evaluating its properties was realized in MATLAB version 6.0 on the computer with processor INTEL Pentium II and with RAM capacity of 64MB. Results may be compared with those in [5].

4. Sumarization of algorithm properties

Here are sumarized advantages and disadvantages of using this algorithm for solving classification tasks. The advantages are:

- Extracted rules are only exact not approximate. This is due to generation them from lower approximations and crisp sets. If there are no identical examples assigned to different classes, a recognition rate of 100% for learning set can be obtained. Misclassification can be easily analyzed by rules.
- Possibility to generate a minimal or a characteristic set of rules.
- Generalization ability can be modified by choosing a suitable value of parameter α. If a test example is out of all created lower approximations or crisp sets in the learning phase, then that example is marked as not classified (but not as misclassified).
- Rule extraction is very fast. Creating lower approximations depends on determining upper approximations and boundary regions. Rules can be obtained from much larger data set in relatively short time.
- The algorithm is simple, comprehensible and easy to programme it.
- The algorithm may be used on data set with continuous and discrete data.

Possible disadvantage is that the generalization ability may be lower in the cases when the characteristics of the learning and testing set are very different.

5. Conclusion

In this paper there is introduced the algorithm for extracting two kinds of rule sets directly from numerical data set based on rough set theory. Rules are defined for lower approximations of sets of objects and crisp sets of objects. The final rule set may be either a minimal or a characteristic one as requested by user. In some applications this algorithm may be more suitable than e.g. neural networks because of its numerous advantages. The described algorithm is simple, comprehensible and easy to programme it. Its implementation works relatively fast and can be applied for solving any classification task.

References

[1] ABE, S. – LAN, M.–S.: A Method for fuzzy rules extraction directly from numerical data and its application to pattern classification. In: *IEEE Transactions on Fuzzy Systems*. Vol. 3, No. 1, February 1995, pp. 18–28.

[2] GRECO, S. – MATARAZZO, B. – SLOWINSKI, R.: Rough Set theory Approach to Decision Analysis. In: *EFDAN´98. Third European Workshop on Fuzzy Decision Analysis and Neural Networks for Management, Planning and Optimization.* Dortmund: June 16.–17., 1998, pp. 1–28.

[3] HENERY, R. J.: Methods for Comparison. In: Mitchie, D. – Spiegelhalter, D. J. - Taylor, C. C.: *Machine Learning, Neural and Statistical Classification*, February 17., 1994, pp. 107-124.

[4] KOMOROWSKI, J. – POLKOWSKI, L. – SKOWRON, A.: *Rough sets: A tutorial.* http://esslli.let.uu.nl/Courses/skowron/skowron.ps. 1999.

[5] KRAWIEC, K. – SLOWINSKI, R. – VANDERPOOTEN, D.: Learning of decision rules from similarity based approximations. In: POLKOWSKI, L. – SKOWRON, A.: *Rough Sets in Knowledge Discovery 2. Applications, Case Studies and Software Systems*. Physica-Verlag, Heidelberg: 1998, pp. 37-54.

[6] MERZ, C. J. – MURPHY, P. M.: UCI Repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html. 1996.

[7] PAWLAK, Z.: Rough Sets Elements. In: POLKOWSKI, L. – SKOWRON, A.: *Rough Sets in Knowledge Discovery 1. Methodology and Applications*. Physica-Verlag, Heidelberg: 1998, pp.10-30.

Contact information:

Pavol Kaločay
Department of Technical Support Systems and Automatization,
Faculty of Air Defence, Military academy, Liptovský Mikuláš, Slovak republic
Telephone: +421/044/5525241, 5525242, E-mail: pavol.kalocay@szm.sk

Michal Turčaník
Department of Informatics and Computer Sciences,
Faculty of Telecommunication Technology and Information Systems, Military academy,
Liptovský Mikuláš, Slovak republic
Telephone: +421/044/5525241, 5525242, E-mail: turcanik@valm.sk