Technical report on Rough Set Theory for Knowlege Discovery in Data Bases

Matteo Magnani *

July 1, 2003

1 Introduction

Knowledge discovery in databases (KDD) is the process of identifying useful information in data. A widely accepted high-level view of this process has been presented in [FPSSU96], where it has been decomposed in a few iterative steps. This model is very simple, but it hides a huge number of different tools, methods and underlying theories used in real-world problems.

One of these is rough set (RS) theory [Paw82], a mathematical formalism developed by Zdzislaw Pawlak to analyze data tables. Its peculiarity is a well understood formal model, which allows to find several kinds of information, such as relevant features or classification rules, using minimal model assumptions.

The aim of this work is to show how rough set theory can be applied to the KDD process. Firstly, we introduce the main aspects of classical rough set theory. In section 3 we show how rough set concepts and methods can be used in the KDD process. Finally, in section 4 we cite some extensions of classical rough set theory and we briefly discuss research problems and trends.

* matteo.magnani@cs.unibo.it - University of Bologna, department of Computer Science

2 Rough set theory

In this section we present the main aspects of classical RS theory. In RS data model information is stored in a table, where each row (tuple) represents a fact or an object. All we know about a real world object is the corresponding tuple in the table.

Interesting data tables are usually difficult to analyze. They store a huge quantity of data, which is hard to manage from a computational point of view. Moreover, it is possible for some facts not to be consistent to each other.

One of the main objectives of RS data analysis is to reduce data size. Section 2.2 introduces the notions of *indiscernibility*, *rough set* and *reduct*, used to approximate inconsistent information and to exclude redundant data. Section 2.3 explains how to measure the quality of the approximations obtained by RS analysis. A more detailed introduction to rough set theory may be found in [KPS99].

2.1 Data model

In RS terminology, a data table is also called an "information system". If some of the attributes are interpreted as outcomes of classification, it is also called a "decision system". An example decision system is represented in Table 1. More formally, a *data table* is a tuple $\langle U, A, V_a \rangle_{a \in A}$ where (see Table 2):

1. U is a set of objects
2. A is a set of attributes $a : U \to V_a$
3. V_a is a set of values for the attribute a.

We can split the set of attributes in two subsets $C \subset A$ and $D = A - C$, respectively the conditional set of attributes and the decision (or *class*) attribute(s). Condition attributes represent measured features of the objects, while the decision attribute is an *a posteriori* outcome of classification. In Table 1 the decision attribute is $CLASS$.

2.2 Dimensionality reduction

A table may be redundant in two ways. The first form of redundancy is easy to notice: some objects may have the same features. This is the case for tuples x_1 and x_2 of Table 1. A way of reducing data size is to store only one representative object for every set of so-called *indiscernible* tuples[1]. The second form of redundancy is more difficult to locate, especially in large data tables. We introduce it using an example. In Table 1 the values of the condition attributes allow us to classify every object. If one banana is 15 centimeter long, smells good, is yellow and comes from field #1, we are sure that it will be sold as fruit. If we do not consider the $SIZE$ attribute, we are no more able to classify objects of type $\langle good, yellow, \#1 \rangle$. In fact, both x_3 and x_4 have these features, but $CLASS(x_3) = FRUIT$ and $CLASS(x_4) = JUICE$. While $SIZE$ is needed to discriminate between different classes, we can remove some other attributes

[1] Notice that we have to remember also the number of tuples in every indiscernibility class to preserve the original information.

#	SIZE	SMELL	COLOR	FIELD	CLASS
x_1	15	good	yellow	#1	FRUIT
x_2	15	good	yellow	#1	FRUIT
x_3	14	good	yellow	#1	FRUIT
x_4	7	good	yellow	#1	JUICE
x_5	12	bad	yellow	#1	JUICE
x_6	13	good	brown	#2	JUICE
x_7	12	bad	brown	#1	REJECT
x_8	14	bad	black	#2	REJECT
x_9	15	bad	black	#2	REJECT

Table 1: A table representing a classification of bananas. $CLASS$ is the decision attribute: bananas may be sold as fruits, used in fruit juices or rejected. The $SIZE$ attribute is expressed in centimeters. Bananas come from two different fields, as stated in the $FIELD$ column.

$$
\begin{aligned}
U &= \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9\} \\
A &= \{SIZE, SMELL, COLOR, FIELD, CLASS\} \\
V_{SIZE} &= \{7, 12, 13, 14, 15\} \\
V_{SMELL} &= \{bad, good\} \\
V_{COLOR} &= \{yellow, brown, black\} \\
V_{FIELD} &= \{\#1, \#2\} \\
V_{CLASS} &= \{FRUIT, JUICE, REJECT\}
\end{aligned}
$$

Table 2: Values for Table 1

#	SIZE	SMELL	COLOR	FIELD	CLASS
x_1	15	good	yellow	#1	JUICE
x_2	15	good	yellow	#1	FRUIT
x_3	14	good	yellow	#1	FRUIT
x_4	7	good	yellow	#1	JUICE
x_5	12	bad	yellow	#1	JUICE
x_6	13	good	brown	#2	JUICE
x_7	12	bad	brown	#1	REJECT
x_8	14	bad	black	#2	REJECT
x_9	15	bad	black	#2	REJECT

Table 3: A modified "banana" table. Notice that tuples x_1 and x_2 have the same values for the condition attributes (they are C-indiscernible), but classes are different. This table is not consistent.

(i.e., columns) without losing classification power. If we erase the $FIELD$ column, we are still able to classify all the objects. More formally,

$$(SIZE(x) = SIZE(x')) \quad \wedge \quad (SMELL(x) = SMELL(x')) \quad \wedge$$
$$\wedge \quad (COLOR(x) = COLOR(x')) \quad \Rightarrow \quad (CLASS(x) = CLASS(x')).$$

In the following sections we show how to manage these two forms of redundancy.

2.2.1 Indiscernibility relation and Rough Sets

As already said, the first form of redundancy is called *indiscernibility*. Two tuples are indiscernible with respect to a set of attributes $B \subseteq A$ if

$$a(x) = a(x') \qquad \forall a \in B$$

It is easy to prove that indiscernibility is an equivalence relation. We denote the B-indiscernibility[2] class of x as $[x]_B$. In Table 1 $[x_1]_C = \{x_1, x_2\}$, where $C = \{SIZE, SMELL, COLOR, FIELD\}$[3]. We will call the partition induced by a B-indiscernibility relation on a set U of objects $IND_B(U)$.

As already pointed out, the indiscernibility relation may be used to represent efficiently sets of tuples. It suffices to keep only one tuple from every indiscernibility class. CX is the representation of the set of tuple X using C-indiscernibility. W.r.t. Table 1, we can represent the set X_{FRUIT}[4] as

$$CX_{FRUIT} = \{[x_1], [x_3]\} \tag{1}$$

It is worth noticing that this is not always possible. A table is said to be *consistent* if every C-indiscernibility class has a unique value for the decision attribute. This means

[2]Relative to attributes in B.

[3]C is the conditional set of attributes.

[4]All tuples with value $FRUIT$ for the decision attribute.

that two tuples which have the same features are equally classified. Whenever a table is consistent, we can always use the above representation.

Unfortunately, not all tables are consistent. For example, in Table 3 tuples x_1 and x_2 are C-indiscernible, but they have different values for the decision attribute. This can happen for several reasons. For example, we may suppose that some features of the objects have not been stored in the table. Or the man who performed the classification may have changed his idea of "good banana". In any case, our description of X_{FRUIT} in (1) is no more valid.

To represent a set which is not precisely definable by indiscernibility classes, we can find a lower and an upper approximations of it, in the following way:

$$\begin{aligned} \underline{C}X_{FRUIT} &= \{x \in U \mid [x]_C \subseteq X_{FRUIT}\} \\ \overline{C}X_{FRUIT} &= \{x \in U \mid [x]_C \cap X_{FRUIT} \neq \emptyset\} \end{aligned}$$

$\underline{C}X$ is the set of indiscernibility classes which are subsets of X. If X is characterized by a particular decision value, this means that all indiscernibility classes in $\underline{C}X$ contain objects with that value. For this reason, data tells us that we are able to classify objects in $\underline{C}X$. There are also indiscernibility classes which contain only some tuples in X. In this case, we cannot classify them. These are the objects in $\overline{C}X - \underline{C}X$, also called *boundary region*. Finally, there may be elements in $U - \overline{C}X$, which contain tuples not in X. Also these can be (negatively) classified. If the boundary region is empty, i.e., $\underline{C}X = \overline{C}X$, X is said to be *crisp* or *precise*. Otherwise, it is called a *rough set*, which in practice is a pair of set approximations.

Rough set theory can be used to represent whichever set X, but we are usually interested in approximating sets of tuples w.r.t. the decision attribute. Image 1 explains graphically what has been said so far about rough approximations.

2.2.2 Reducts

As we have seen in section 2.2, some columns of a table may be erased without affecting the classification power of the system. This concept can be extended also to tables where we do not distinguish between conditional and decision attributes. In this case, an attribute can be removed if there are no two tuples which become indiscernible. A *reduct* is a minimal set of attributes that preserve the indiscernibility relation.

More formally, given a table $\langle U, A, V_a \rangle$, a reduct is a set of attributes R such that:

- $R \subseteq A$
- $IND_R(U) = IND_A(U)$
- $IND_{R-a}(U) \neq IND_A(U) \quad \forall a \in R$

While computing equivalence classes is easy, finding minimal reducts is NP-hard. An interesting approach to solve this problem, which in practice is tackled using strong heuristics, is based on Boolean reasoning. Given a table, for every pair of tuples we list the attributes which are different. Table 4 is a more realistic example of classification of bananas, which has already been reduced by using indiscernibility classes.

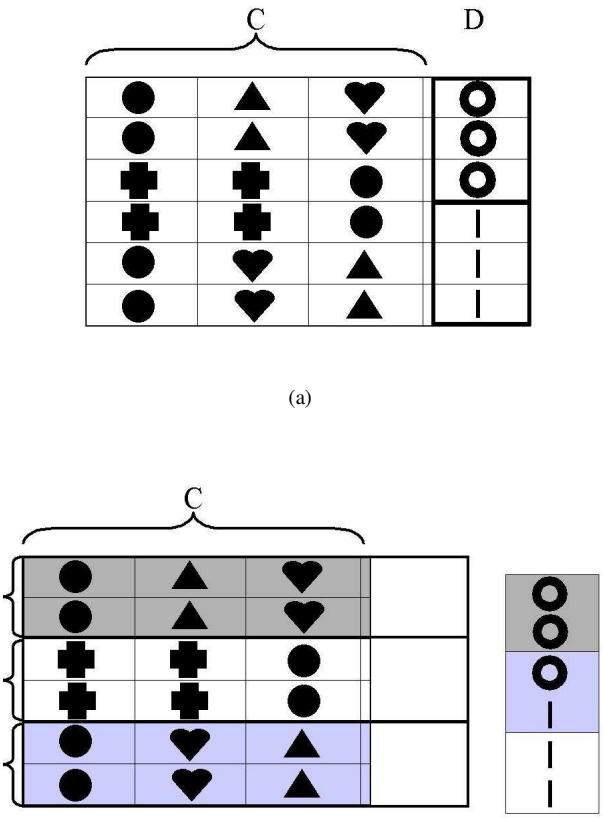

Figure 1: This figure shows the approximations of the objects classified with the symbol **O**. 1(a) shows that tuples 1, 2 and 3 (from the top) belong to the same class. In 1(b), C-indiscernible tuples have been collected in three regions, corresponding to the C-indiscernibility classes. Figure 1(c) illustrates the approximations of class **O**. The first region from the top is the lower approximation. The union of the top two regions is the upper one.

#	SIZE	SMELL	COLOR	FIELD	CLASS	NUM
CL_1	14	good	yellow	#1	FRUIT	349
CL_{2a}	15	good	yellow	#1	FRUIT	347
CL_{2b}	15	good	yellow	#1	JUICE	12
CL_3	7	good	yellow	#1	JUICE	145
CL_4	12	bad	yellow	#1	JUICE	18
CL_5	13	good	brown	#2	JUICE	254
CL_6	12	bad	brown	#1	REJECT	32
CL_7	14	bad	black	#2	REJECT	147
CL_8	15	bad	black	#2	REJECT	163

Table 4: This is a more realistic instance of our example. Tuples have been aggregated in C-indiscernibility classes. Class CL_2 is not consistent, so it has been subdivided into two classes with different values for the decision attribute. The NUM column represents the number of tuples in the corresponding indiscernibility class.

classes	CL_2	CL_3	CL_4	CL_5	CL_6	CL_7	CL_8
CL_1	SI	SI	SI SM	SI C F	SI SM C	SM C F	SI SM C F
CL_2		SI	SI SM	SI C F	SI SM C	SI SM C F	SM C F
CL_3			SI SM	SI C F	SI SM C	SI SM C F	SI SM C F
CL_4				SI SM C F	C	SI C F	SI C F
CL_5					SI SM F	SI SM C	SI SM C
CL_6						SI C F	SI C F
CL_7							SI

Table 5: Indiscernibility matrix for Table 4. This matrix is always symmetric. For this reason, only the upper half has been represented. Entry $(CL_x, CL_{x'}) = a_1, \ldots, a_k$ corresponds to the expression $a_1 \cup \ldots \cup a_k$.

For every pair of tuples, we can list all attributes which allow us to discern between them. Table 5 represents the so-called indiscernibility matrix for Table 4. Every entry $(CL_x, CL_{x'}) = a_1, \ldots, a_k$ of the matrix corresponds to a Boolean expression $E(CL_x, CL_{x'}) = a_1 \cup \ldots \cup a_k$. If we want to know if two tuples are discernible using only some of the attributes, we can do the following. We assign *true* to the attributes we are considering and *false* to the remaining ones, then we evaluate the corresponding expression. $E(CL_x, CL_{x'})$ tells us if we are able to discern CL_x from $CL_{x'}$.

If we want to know when *all* objects are discernible from each other, we can take the conjunction of all entries of the indiscernibility matrix. In the example, we obtain the following so-called indiscernibility function:

$$f(SI, SM, C, F) =$$
$$(SI)(SI)(SI \cup SM)(SI \cup C \cup F)(SI \cup SM \cup C)(SM \cup C \cup F)(SI \cup SM \cup C \cup F)$$
$$(SI)(SI \cup SM)(SI \cup C \cup F)(SI \cup SM \cup C \cup)(SI \cup SM \cup C \cup F)(SM \cup C \cup F)$$

#	SIZE	COLOR	CLASS	NUM
CL_1	14	yellow	FRUIT	349
CL_{2a}	15	yellow	FRUIT	347
CL_{2b}	15	yellow	JUICE	12
CL_3	7	yellow	JUICE	145
CL_4	12	yellow	JUICE	18
CL_5	13	brown	JUICE	254
CL_6	12	brown	REJECT	32
CL_7	14	black	REJECT	147
CL_8	15	black	REJECT	163

Table 6: Table 4 after reduction. Notice that all tuples are still discernible from each other.

$$\begin{aligned}(SI \cup SM)(SI \cup C \cup F)(SI \cup SM \cup C\cup)(SI \cup SM \cup C \cup F)(SI \cup SM \cup C \cup F)\\ (SI \cup SM \cup C \cup F)(C)(SI \cup C \cup F)(SI \cup C \cup F)\\ (SI \cup SM \cup F)(SI \cup SM \cup C)(SI \cup SM \cup C)\\ (SI \cup C \cup F)(SI \cup C \cup F)\\ (SI) \quad (2)\end{aligned}$$

This seems a complex equation, but it is very easy. In substance, it is another way of writing the indiscernibility matrix. Given a truth assignment corresponding to the variables we want to keep, every group of disjunctions is true iff we are able to distinguish between two particular indiscernibility classes. For example, assume we want to use only the $SMELL$ attribute. $E(CL_1, CL_2) = (SI)$ is false, so we cannot discern class CL_1 from CL_2. In fact, $SMELL(CL_1) = SMELL(CL_2) =$ *good*. On the contrary, $E(CL_1, CL_4) = (SI \cup SM)$ is true, so we can discern class CL_1 from CL_4. In fact, $SMELL(CL_1) =$ *good* while $SMELL(CL_4) =$ *bad*. Equation (2) is true iff all groups of disjunctions are true. This means that we are able to discern any two classes from each other using only the attributes whose corresponding variables are true.

The problem of finding a subset of the attributes which preserve the indiscernibility relation can be reduced to the problem of finding the implicants of equation (2). An implicant is a conjunction of variables such that if these variables are true also the function is true. In particular, we are mainly interested in prime implicants, which are implicants of minimal size.

Equation (2) can be reduced to $f(SI, SM, C, F) = SI \cap C$. This means that attributes $SIZE$ and $COLOR$ are sufficient to preserve the indiscernibility relation. Table 6 represents the smallest possible redundancy reduction using only information from data.

2.3 Numerical measures of rough approximations

We conclude this section by introducing some numerical measures which characterize RS approximations.

It is worth noticing that these measures belong to the classical rough set theory, but they are the first step towards extensions of this model. The main idea of rough approximations is that a set can be represented by a lower approximation and an upper one. Now, assume that an indiscernibility class have 1000 elements. If 999 of these have been classified in X, this set belongs to the boundary region of X. If only one has been classified in X, it is treated in the same way. This example shows that two approximations are not always sufficient to represent sets of objects in a satisfactory way.

Accuracy measures how much a set is rough. If a set has $\underline{C}X = \emptyset$ and $\overline{C}X = U$, the approximation tells us nothing about X, because for any element $x \in U$ we cannot decide if $x \in X$. On the contrary, if $\underline{C}X = \overline{C}X = X$, the set is crisp (precise) and for every element $x \in U$ we know if $x \in X$ or not. This is expressed by the following formula:

$$\alpha_C(X) = \frac{|\underline{C}X|}{|\overline{C}X|} \tag{3}$$

It is easy to notice that $0 \leq \alpha_C(X) \leq 1$, and if $\alpha_C(X) = 1$ X is crisp with respect to C.

Rough membership tells us how much an indiscernibility class belongs to a set X. It is defined in the following way:

$$\mu_X^C(x) = \frac{|[x]_C \cap X|}{|[x]_C|} \tag{4}$$

When class $[x]_C$ is a proper subset of X, $\mu_X^C(x) = 1$. Otherwise $0 \leq \mu_X^C(x) < 1$.

3 Rough sets and Knowledge Discovery in Databases

Given the complexity and the task-dependence of the KDD process, it is difficult to decompose it into elementary steps. In this section we use the following decomposition, based on the well-known description in [FPSSU96]:

1. Understanding the domain, prior knowledge and goals.
2. Creating a target data set.
3. Data cleaning and preprocessing.
4. Data Mining.
5. Interpretation/Evaluation.

Following the scheme used in [DGN], in next sections we identify possible applications of RS theory in these subprocesses. Steps 1 and 2 are not supported by RS data analysis.

3.1 Data cleaning and preprocessing

Data cleaning and preprocessing is one of the application fields of rough sets. It is known that prior knowledge is very important in the KDD process. On the other side, one of the main features of rough set data analysis is that it does not use information outside the target data set[5]. These two things seem to be incompatible. A good compromise can be the following. At first, rough sets are used to reduce and clean data with minimal model assumptions. Then, the result is used as a basis for further analysis performed with other methods.

In the context of data cleaning and preprocessing, rough set theory may be useful to handle the following problems:

- Data reduction.
- Missing value handling.
- Feature Selection.
- Feature Extraction.

3.1.1 Data reduction

Indiscernibility relation and reducts are the tools used to reduce data size. Table 6 shows how more than 1400 tuples may be represented in a compact way using only two conditional attributes.

[5]In some sense, this is also prior knowledge. Data tables are constructed in a particular way because experts decide which features are important, based on their understanding of the problem or on previous model assumptions. When we say that rough set data analysis does not use prior knowledge, we refer to external subjective parameters such as prior probabilities or fuzzy membership functions.

Unfortunately, looking more carefully at this example it is easy to notice some serious problems. The indiscernibility relation defined in section 2 is very strict[6], so in tables with many features we can possibly have an indiscernibility class for every tuple. Additionally, look at classes CL_7 and CL_8. We would say that they are *quite* indiscernible. If some attributes have continuous domains, it is very difficult to find large and useful equivalence classes, even with a few features. As a consequence, the problem of numerical attribute values can also influence the search for reducts. In our example we have erased the $SMELL$ column because it was not necessary to distinguish between classes, e.g., CL_5 and CL_6. If we group together bananas which are 12 or 13 centimeter long, which is reasonable, the $SMELL$ attribute is necessary to classify them correctly.

To face these problems, tolerance relations have been introduced[7]. Tolerance relations include in the same class tuples which are similar. This is very useful also to reduce noise, but implies the use of a metric to compare objects, which is a concept outside standard RS theory.

Technically, a tolerance relation on U is a subset of $U \times U$ which is reflexive and symmetric, but not necessarily transitive. In [SV00] also the symmetry constraint is relaxed.

3.1.2 Missing value handling

Missing values can be handled before RS data analysis. We can use some data reparation methods from KDD, to obtain a complete system. In [GBH01] nine of them are compared. A main drawback of these methods is that the statistical distribution of the attribute values is not usually known *a priori*.

Otherwise, traditional RS methods may be extended to manage *null* values. There are several ways to do this, all based on tolerance relations. In [Wan] a tolerance relation is presented, which classifies together objects with some missing values. In [Val03] the author does something similar, but introducing a third approximation for objects which are not similar but have *null* values. Both these approaches do not answer completely to this problem.

3.1.3 Feature Selection and Extraction

Feature selection can be performed in several ways using RS theory. Some of these are described in [KPS99] and [DG00]. One method is to evaluate all reducts and to take the so-called *core*, that is the intersection of all of them. The core is very simple to evaluate. Another possibility is to divide the data table into subtables and keep only the reducts which appear *sufficiently often*[8]. These are called *approximate reducts*, and may be useful to reduce noise effects.

These methods distinguish between *useful* and *useless* attributes. In reality, every attribute is more or less useful, and we can define a continuous measure to quantify

[6] Tuples must be equal for all attributes.

[7] In particular, to obtain non-singleton indiscernibility classes. Discretization and grouping are problems for which a general and completely satisfactory solution does not exist.

[8] Obviously, this parameter must be evaluated numerically.

it. This is done in two steps: firstly, we evaluate how much decision attributes depend on condition ones. Then, we erase an attribute and we evaluate again the dependency. The more it is important for classification, the more this value will decrease. From the difference of dependency, we obtain a value for the utility of the attribute.

The first step is to evaluate dependency between attributes. It is defined in the following way:

$$\gamma(B, D) = \frac{\sum_{X \in IND_D(U)} |\underline{B}X|}{|U|}$$

If all sets X are crisp, $\gamma(B, D) = 1$. In fact, this means that using the attributes in B we can precisely define the partition $IND_D(U)$. If we cannot precisely decide about the membership of any object, that is $\underline{B}X = \emptyset, \gamma(B, D) = 0$

Then, we calculate how much removing an attribute changes the original dependency.

$$\sigma_{B,D}(a) = \frac{\gamma(B, D) - \gamma(B - \{a\}, D)}{\gamma(B, D)}$$

If $B - \{a\}$ is a reduct, $\gamma(B, D) = \gamma(B - \{a\}, D)$ and $\sigma_{B,D}(a) = 0$. In fact, the attribute a is not significant at all. If B is a reduct, it may happen that some of its attributes are less useful then others, because they allow to distinguish between a few small classes.

Discretization is a main topic for many data analysis tools, and also for RS methods. In particular, it is possible to find discretizations which preserve decision classes *via* Boolean reasoning. This is done in the following way:

1. For every attribute, all possible *cuts* are listed. A cut is the median point between two adjacent values. For example, if an attribute takes the values $\{1, 3, 6, 9, 11\}$, cuts are $\{2, 4.5, 7.5, 10\}$.

2. For every pair of objects with different decision attributes, the set of cuts which distinguishes them is identified.

3. These cuts are used to evaluate a discernibility formula, as we have done to find reducts.

We show an example of this procedure (see Table 7):

1. First of all, we enumerate all attribute values and all cuts:
   $V_{A1} = \{1, 3, 5, 7, 8, 9\}, V_{A2} = \{1, 2, 3, 4\}$
   $Cuts_{A1} = \{2, 4, 6, 7.5, 8.5\}, Cuts_{A2} = \{1.5, 2.5, 3.5\}$

#	A1	A2	D
a_1	1	2	YES
a_2	5	3	YES
a_3	3	4	NO
a_4	8	1	YES
a_5	9	1	NO
a_6	7	4	NO

Table 7: Attributes of this table have continuous values.

#	A1	A2	D
a_1	Ca1_1	Ca2_1	YES
a_2	A1_1	A2_1	YES
a_3	A1_1	A2_2	NO
a_4	A1_1	A2_1	YES
a_5	A1_2	A2_1	NO
a_6	A1_1	A2_2	NO

Table 8: Table 7 after discretization. For every attribute, we have selected one cut. This divides the continuous domain of the attribute in two discrete spaces.

2. Then we list the cuts which distinguish tuples from different classes:

$$E(a_1,a_3) = \underbrace{2,}_{A1}\underbrace{2.5,3.5}_{A2} \qquad E(a_1,a_5) = \underbrace{2,4,6,7.5,8.5,}_{A1}\underbrace{1.5}_{A2}$$
$$E(a_1,a_6) = \underbrace{2,4,6,}_{A1}\underbrace{2.5,3.5}_{A2} \qquad E(a_2,a_3) = \underbrace{4,}_{A1}\underbrace{3.5}_{A2}$$
$$E(a_2,a_5) = \underbrace{6,7.5,8.5,}_{A1}\underbrace{1.5,2.5}_{A2} \qquad E(a_2,a_6) = \underbrace{6,7.5,8.5,}_{A1}\underbrace{1.5}_{A2}$$
$$E(a_4,a_3) = \underbrace{4,6,7.5,}_{A1}\underbrace{1.5,2.5,3.5}_{A2} \qquad E(a_4,a_5) = \underbrace{8.5}_{A1}$$
$$E(a_4,a_6) = \underbrace{7.5,}_{A1}\underbrace{1.5,2.5,3.5}_{A2}$$

For example, $E(a_4,a_5) = \underbrace{8.5}_{A1}$. In fact, points $a_4 : (8,1)$ and $a_5 : (9,1)$ can be separated by the value 8.5 for the first attribute. This is clear looking at Figure 2.

3. One prime implicant of the corresponding indiscernibility function is $\underbrace{8.5,}_{A1}\underbrace{3.5}_{A2}$.

Figure 2 illustrates the corresponding cuts. Table 8 shows the new table obtained by discretizing Table 7.

This method is useful to reduce data size, but it does not introduce new information in the table. Useful discretizations should also put together objects from different

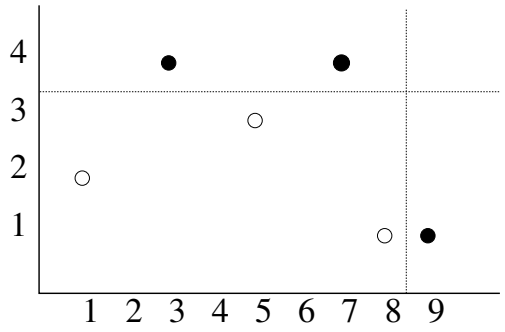

Figure 2: Graphical representation of Table 7. Black circles represent the objects in class NO, white circles the ones in class YES. Only two values per attribute are needed to distinguish the two classes. Discretization obtained by Boolean reasoning is illustrated by dotted lines.

classes, when the different classification is due to other attributes. For example, in table 4 attributes with $SIZE = 12$ and $SIZE = 13$ should be put together, but they are not using this method, because CL_5 and CL_6 belong to different classes.

3.2 Data Mining

"Data mining" indicates a family of different tasks. Between them, rough set theory has been used within the following:

- Classification.
- Clustering.
- Association rules.

3.2.1 Classification and clustering

Classification is the original goal of rough set theory. We may distinguish two cases: when the information table is consistent and when it is inconsistent.

In the former case, we may calculate the reducts of the table and use them in the obvious way. For example, from Table 6 we can extract the rule:

$$\text{if} \quad size = 14 \quad \text{and} \quad color = black \quad \text{then} \quad reject.$$

This approach is very simple, but it cannot be applied to inconsistent information. Furthermore, we risk over-fitting, with a resulting poor classification power for new unseen objects. A better idea can be to find only *approximate (dynamic) reducts*, which have been defined in section 3.1.3. In this way we can preserve *quite well* the indiscernibility relation, reducing noise effects. However, we have the additional problem of defining the numerical meaning of *quite well*.

When a system is not consistent, we may generate rules with certainty factors. When new objects must be classified, a voting procedure decides the rule to be fired. In [WL01] an algorithm is proposed, which improves another one presented by Skowron. Anyway, this is not a definitive answer to this problem, because it does not consider important topics such as tolerance relations, missing value handling, support and significance of the rules.

At the beginning, RS theory was used mainly to classify objects, i.e., to assign them to classes known *a priori*. Recently, some researcher has also tried to apply it to clustering. Roughly speaking, clustering is a form of classification where we do not know which are the classes. As in classification, real problems can present overlapping classes. When this is due to data coarseness, rough set techniques may be used. In [LY02] two algorithms are compared from a computational point of view, one based on genetic algorithms and a modified version of K-means.

Finally, rough sets attribute dependency analysis may be used to quantify numerically association rules, on the basis of information in data. Anyway, there is no much literature on this topic.

4 Final remarks

In our opinion, classical RS theory was interesting because of two main reasons. The former was the idea of approximating an inconsistent set with two consistent ones, which is very elegant. The latter was its mathematical formulation, simple and formal at the same time. It allows to reduce important problems[9] to well-known Boolean ones, using minimal model assumptions (i.e., only data in tables).

So far, RSs have had only a partial success. There are a lot of publications on the argument, and it has been applied to several research areas ([KPS99, 44–47]). Recently, it has been proposed as a basis for granular computing ([Lin03]). However, in [Zia01] the author points out that there are a lot of experimental applications, but industrial use is rare. Moreover, there are no specialized journals on this topic. While other important formalisms, such as fuzzy sets, have been adopted by several researchers in different fields, RS theory is mainly used only inside its research community.

Probably, the two above reasons of interest are no well suited to face real problems. In some sense, classical RS theory is *too elegant*! In real world situations, collected data is usually uncertain and incomplete. For this reason, two approximations are not sufficient to properly describe the information hidden in data.

To deal with complex data, the theory has been adjusted. The indiscernibility relation has been substituted by tolerance relations ([SV00], [ŚZ03]), useful in presence of many features and missing values. Rough membership functions have been parameterized to manage noisy data ([Zia93]) or extended to more approximations to deal with incomplete information ([Val03]). Finally, hybridizations with other methods have been found, such as fuzzy sets or neural networks.

With these changes, RS theory is probably more suited to analyze real world data, but it has lost its main features. How do we justify its use, when other methods already exist? Are there new contributions? In [DG] the authors claim that (one of) the most important feature of RS data analysis is its objectivity:

> Rough set analysis uses only internal knowledge, and does not rely on prior model assumptions as fuzzy set methods or probabilistic models do.

The problem is that data tables are representations of real-world data filtered by prior assumptions. Every attribute of a table has been added because someone considered it interesting. And this is not very different from specifying also a degree of importance.

If we let prior knowledge influence data tables, we substitute lower and upper approximations with rough membership functions, and we introduce external parameters, the result is something very similar to fuzzy sets, evidence theory, and other techniques of automated learning. In fact, in [YZ01] the authors use RS theory to interpret fuzzy membership functions, and in [KPS99, pp.36–44] relationships with other approaches are discussed, such as Shafer's evidence theory. Other successful methods have been accepted by the scientific community because they had a modeling power different from that of existing theories. Therefore, it would be useful to stress more the differences between these approaches, identifying situations that we can straightforwardly model only using rough sets.

[9] Such as the search for reducts.

References

[DG] I. Düntsch and G. Gediga. Roughian - rough information analysis. URL: citeseer.nj.nec.com/488866.html.

[DG00] I. Düntsch and G. Gediga. Rough set data analysis, 2000. URL: citeseer.nj.nec.com/dntsch00rough.html.

[DGN] I. Düntsch, G. Gediga, and H.S. Nguyen. Rough set data analysis in the kdd process. URL: citeseer.nj.nec.com/387773.html.

[FPSSU96] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors. *Advances in knowledge discovery & data mining*, chapter 1, pages 1–36. MIT Press, Cambridge, MA, 1996.

[GBH01] J.W. Grzymala-Busse and M Hu. A comparison of several approaches to missing values in data mining. In Ziarko and Yao [ZY01].

[Jär01] J. Järvinen. Approximations and rough sets based on tolerances. In Ziarko and Yao [ZY01].

[KPS99] J. Komorowski, L. Polkowski, and A. Skowron. Rough sets: a tutorial. In S.K. Pal and A. Skowron, editors, *Rough-Fuzzy Hybridization*, pages 3–98. Springer–Verlag, 1999. citeseer.nj.nec.com/komorowski98rough.html.

[Lin03] T.Y. Lin. Granular computing - structures, representations and applications. In Wang et al. [WLYS03].

[LY02] P. Lingras and Y.Y. Yao. Time complexity of rough clustering: Gas versus k-means. In James J. Alpigini, James F. Peters, Jacek Skowronek, and Ning Zhong, editors, *Rough Sets and Current Trends in Computing*, volume 2475 of *Lecture Notes in Computer Science*. Springer, 2002.

[Paw82] Z. Pawlak. Rough sets. *Int. J. Computer and Information Sci.*, 11:341–356, 1982.

[SV00] R. Slowinski and D. Vanderpooten. A generalized definition of rough approximations based on similarity. *IEEE Transactions on Data and Knowledge Engineering*, 12(2):331–336, 2000.

[ŚZ03] D. Ślęzak and W. Ziarko. Variable precision bayesan rough set model. In Wang et al. [WLYS03].

[Val03] E. Valentini. Linguaggio di interrogazione per basi di dati multimediali basato su fuzzy set e rough set. Master's thesis, University of Bologna, 2003.

[Wan] Guoyin Wang. Extension of rough set under incomplete information systems.

[WL01] G. Wang and F. Liu. The inconsistency in rough set based rule generation. In Ziarko and Yao [ZY01].

[WLYS03] G. Wang, Q. Liu, Y. Yao, and A. Skowron, editors. *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing 9th International Conference, RSFDGrC 2003, Chongqing, China, May 26-29, 2003. Proceedings*, volume 2639 of *Lecture Notes in Artificial Intelligence*. Springer, 2003.

[YZ01] Y.Y. Yao and J.P. Zhang. Interpreting fuzzy membership functions in the theory of rough sets. In Ziarko and Yao [ZY01].

[Zia93] W. Ziarko. Variable precision rough sets model. *Journal of Computer and Systems Sciences*, 46(1):39–59, 1993.

[Zia01] W. Ziarko. Rough sets: Trends, challenges and prospects. In Ziarko and Yao [ZY01].

[ZY01] Wojciech Ziarko and Y. Y. Yao, editors. *Rough Sets and Current Trends in Computing, Second International Conference, RSCTC 2000 Banff, Canada, October 16-19, 2000, Revised Papers*, volume 2005 of *Lecture Notes in Computer Science*. Springer, 2001.