Rough set-based rule generation and Apriori-based rule generation from table data sets II: SQL-based environment for rule generation and decision support

Hiroshi Sakai, Zhiwen Jian

Abstract: This study follows the previous study entitled ‘Rough set-based rule generation and Apriori-based rule generation from table data sets: A survey and a combination’, and this is the second study on ‘Rough set-based rule generation and Apriori-based rule generation from table data sets’. The theoretical aspects are described in the previous study, and here the aspects of application, an SQL-based environment for rule generation and decision support, are described. At first, the implementation of rule generator defined in the previous study is explained, then the application of the obtained rules to decision support is considered. Especially, the following two issues are focused on, (i) Rule generator from table data sets with uncertainty in SQL, (ii) The manipulation in decision support below: (ii-a) In the case that an obtained rule matches the condition, (ii-b) In the case that any obtained rule does not match the condition. The authors connect such cases with decision support and realised an effective decision support environment in SQL.

1 Introduction

Recently, there are several table data sets that store instances of previous occurrences. Data mining techniques allow us to investigate instances in table data sets, and we know trends and characteristics of data sets. In addition, rule-based decision support seems to be a very active research field. In fact, we found over 7700 papers from the Scopus system [1] by a keyword ‘rule-based decision support’. The Scopus system also responded the amount of papers in some research areas, and we had the following composition ratios, 35% in computer science, 24% in engineering, 13% in medicine, and 11% in mathematics, 5% for decision-making science, 5% for social science, 4% for business and management, 3% for biological science etc. In these papers, fuzzy sets and rough sets seem to be very important.

Several fuzzy frameworks are proposed in [2–8], and rough set-based frameworks are also proposed in [9–14]. The authors in this paper also developed a rough set-based framework, and termed it rough sets non-deterministic information analysis (RNIA) [15–17]. RNIA is a framework for handling not only tables with definite information (which we term DISs: Deterministic Information Systems) but also tables with indefinite information (which we term NISs: Non-deterministic Information Systems). NIS was proposed by Pawlak [12] and Orłowska and Pawlak [18] for handling information incompleteness.

When NIS was proposed, the research purpose was to establish effective question-answering from tables with information incompleteness. However, we are here focusing on rule generation from NISs and decision support based on the obtained rules. RNIA framework maintains logical aspects. In other words, the core rule generation algorithm termed the NIS-Apriori algorithm [16, 19] is sound and complete [20, 21] for minimal rules. Therefore, this algorithm does not miss any minimal rule. Any minimal rule is certainly generated by this algorithm. This is assured theoretically. There seems less data mining algorithm with such logical properties.

In the RNIA framework, we define certain rules and possible rules based on possible world semantics [20, 21]. These definitions seem to be natural and standard, but the amount of possible worlds may exceed $10^{100}$ in the Mammographic data set and the Congressional Voting data set [22]. It will be impossible to examine the constraints depending upon more than $10^{100}$ cases sequentially. Even though this examination seems to be difficult, the NIS-Apriori algorithm provides a solution to this problem. Namely this algorithm is independent from the amount of the possible worlds [15, 16]. Without such property, it will be hard to compute rules defined by the possible world semantics.

The main issue in this paper is to propose three phases (i), (ii), (iii) in Figs. 1 and 2 and its prototype system in SQL.

(i) Rule generation phase: We specify two threshold values $\alpha$ and $\beta$ and generate rules. We will know the trends and characteristics of the data set. Theoretical discussion and the novelty about rule generation phase are described in the twinned paper [23], and we trace rule generation process by using the Lenses data set in UCI machine-learning repository [22] for showing the overall flow.

(ii) Search phase for the obtained rules: For the specified condition part, if a rule matches the specified condition part, we have the rule’s decision part as the decision. (Since we usually consider $\beta > 0.5$ as a constraint, we have one decision for the specified condition part.)

(iii) Question-answering phase: If there is no rule with the same condition part, all implications with the condition part are searched, and we conclude one decision from the searched results with support$(\tau)$ and accuracy$(\tau)$.

If phase (ii) is applicable to the specified condition part, the execution of decision making is much faster than the execution in phase (iii). Therefore, it is useful to apply phase (ii), but there is a possibility that there is no rule which matches the specified condition part. Therefore, it is necessary to prepare phase (iii). Phase (iii) may take time to execute, but this phase responds all implications with the specified condition part and support$(\tau)$, accuracy$(\tau)$ values. That is all information for decision making from the table data set.

This paper is twinned with the previous paper entitled ‘Rough set-based rule generation and Apriori-based rule generation from table data sets: A survey and a combination’ [23]. Thus, in order
to avoid the duplication, we may omit some definitions and issues in this paper. We will mainly describe the rule generator in SQL and the application of the obtained rules to decision support. This paper is organised as follows: Section 2 describes rules and the rule generator in DISs, and Section 3 describes rules and the rule generator in NISs. Section 4 clarifies the problem on decision support functionality in DISs and NISs, and investigates decision support procedures in SQL. Section 5 concludes this paper.

2 Rules from DISs and rule generator in SQL

This section focuses on rule generation from DISs. We especially handle the Lenses data set as an example of DIS.

2.1 Rules from DISs

The Lenses data set stores the symptoms on eyes and diagnosis for patients. This table data set consists of 24 objects (tuples), four attributes: age, spec, asti, and tear, one decision attribute class with three attribute values 1 (hard contact lenses), 2 (soft contact lenses), and 3 (no contact lenses).

Each attribute value can be regarded as a classified value; so, it may be difficult to consider the variance of the average value or statistics. With such a table data set, we will consider rule-based decision support. The Lenses data set in Fig. 3 is an example of DIS, and each attribute value is one value. There is no missing value [24].

A pair \([A, val_A]\) (an attribute \(A\) and its attribute value \(val_A\)) is called a descriptor. For the decision attribute \(Dec\) and a set \(CON\) of condition attributes, we term a formula \(\tau \wedge \lambda_{\neg \text{CONST}} [A, val_A] \Rightarrow \) as a condition.

![Fig. 1 Chart of three phases in decision support in DIS](image1)

![Fig. 2 Chart of three phases in decision support in NIS](image2)

[Dec, val] an implication. For specified threshold values \(0 < \alpha, \beta \leq 1\), we say \(\tau\) is (a candidate of) a rule, if \(\tau\) satisfies the next two criterion values [6, 10, 13, 25].

\[
\begin{align*}
(1)\text{support}(\tau) &= \frac{|eq(\lambda_{\neg \text{CONST}} [A, val_A] \wedge \{\text{Dec}, \text{val}\})|}{|OB|} \geq \alpha, \\
(2)\text{accuracy}(\tau) &= \frac{|eq(\lambda_{\neg \text{CONST}} [A, val_A] \wedge \{\text{Dec}, \text{val}\})|}{|eq(\lambda_{\neg \text{CONST}} [A, val_A])|} \geq \beta.
\end{align*}
\]

Here, \(OB\) is a set of objects, \(eq(\cdot)\) is an equivalence class defined by the formula \(\cdot\), and \(|\cdot|\) is the cardinality of a set \(\cdot\). If \(|eq(\lambda_{\neg \text{CONST}} [A, val_A])| = 0\), we define both \(\text{support}(\tau) = 0\) and \(\text{accuracy}(\tau) = 0\). For an implication \(\tau_1: [\text{asti}, 2] \Rightarrow [\text{class}, 3]\) in Fig. 4,

\[
\begin{align*}
|eq(\tau_1)| &= 8, |eq([\text{asti}, 2])| = 12, \\
\text{support}(\tau_1) &= 8/24 \equiv 0.33, \text{accuracy}(\tau_1) = 8/12 \equiv 0.67. \quad (1)
\end{align*}
\]

Since \(\tau_1\) is located in the rule area, we see this \(\tau_1\) is a candidate of a rule. Similarly, for an implication \(\tau_2: [\text{asti}, 2] \wedge [\text{spec}, 2] \Rightarrow [\text{class}, 1],\)

\[
\begin{align*}
|eq(\tau_2)| &= 1, |eq([\text{asti}, 2] \wedge [\text{spec}, 2])| = 6, \\
\text{support}(\tau_2) &= 1/24 \equiv 0.04, \text{accuracy}(\tau_2) = 1/6 \equiv 0.17. \quad (2)
\end{align*}
\]

As for \(\tau_2\), it is not located in the rule area; so, we see it is not a candidate of a rule. The \(\text{support}(\tau)\) value means the occurrence ratio of the implication \(\tau\). If \(\tau\) occurs frequently, this \(\tau\) is reliable. On the other hand, the \(\text{accuracy}(\tau)\) value means the consistency ratio of \(\tau\). If the \(\text{accuracy}(\tau)\) value is higher, this \(\tau\) is more reliable, too.

2.2 Rule generator in SQL for DISs

We employed the DIS-Apriori algorithm, which is closely described in the twinned paper. In this sub-section, we trace the following procedure \(apriv2\) by using the Lenses data set. This trace will be useful explanation of rule generation.

In each table data set, the number of attribute values and the names of attribute are different. We at first translate each table to a table of the rdf format [26–28]. Fig. 5 shows a part of the rdf table. In this rdf table, each attribute name is handled as a value instead of attribute name. In SQL procedure, we need to specify conditions by using attribute names. So, if we employ the original table directly, the procedure depends upon such attribute names. Namely, we need one procedure for each original table. However, in the rdf table we can employ only three fixed attribute names, i.e. object, attrib, and value. Actually, we implemented the DIS-Apriori algorithm in SQL by using this rdf table. We termed this procedure \(apriv2\) and

![Fig. 4 Rules (defined by support ≥ 0.3 and accuracy ≥ 0.6) plotted in the plane](image3)

![Fig. 3 Some instances of the Lenses data set](image4)
DIS_Step 1) Generation of tables con1, deci, rule1, and rest1. In the table rule1, each rule in the form of \([A, a] \Rightarrow \{\text{Dec, val}\}\) is stored.

DIS_Step 2) Generation of tables con2, rule2, and rest2. In the table rule2, each minimal rule in the form of \([A, a] \land [B, b] \Rightarrow \{\text{Dec, val}\}\) is stored.

DIS_Step 3) Generation of tables con3, rule3, and rest3. In the table rule3, each minimal rule in the form of \([A, a] \land [B, b] \land [C, c] \Rightarrow \{\text{Dec, val}\}\) is stored.

In DIS_Step 1, apriv2 generates the table con1 in Fig. 7. The condition descriptors satisfying \(\text{support} \geq \alpha\) are stored in the table con1. The total number \(|\text{OB}| = 24\) and \(\alpha = 0.2\) are specified; so, if a descriptor occurs more than five \((\geq 24 \times 0.2 = 4.8)\) times, this descriptor satisfies the constraint \(\text{support} \geq \alpha\). Thus, apriv2 handles such a descriptor, and we say such a descriptor is a considerable descriptor. The apriv2 procedure ignores other descriptor whose occurrence is less than four times. Actually, the descriptor \([\text{class}, 1]\)] occurs four times, and \([\text{class}, 1]\)] is not stored in the table deci. Therefore, we have no rule whose decision part is \([\text{class}, 1]\).

The apriv2 procedure combines each descriptor in tables con1 and deci, and examines \(18 (=9 \times 2)\) implications. If an implication \(\tau\) satisfies the constraint, this \(\tau\) is added to the table rule1. If \(\tau\) satisfies \(\text{support}(\tau) \geq 0.2\) and \(\text{accuracy}(\tau) < 0.6\), this \(\tau\) is added to the table rest1 in Fig. 8. We say such an implication is a considerable implication.
In DIS_Step 2, apri2 picks up considerable implications in rest1 and generates considerable implications in con2. Similarly, apri2 generates two tables rule2 and rest2 in Fig. 9. Since rest2 is empty, the rule generation process substantially finishes in this step.

As we have described in the twinned paper, there is no missing minimal rule by the procedure apri2. We had totally 6 implications satisfying support(τ) ≥ 0.2 and accuracy(τ) ≥ 0.6. Fig. 10 is a part of apri2 in SQL. This part generates the table rule1.

3 Rules from NISs and rule generator in SQL

This section focuses on rule generation from NISs. We especially handle the Credit Approval data set in UCI machine learning repository [22] as an example of NIS.

3.1 Rules from NISs

This table data set consists of 690 objects (tuples), 15 attributes: a1, a2, ..., a15, one decision attribute a16 with two attribute values 1 and 2.

In the Credit Approval data set, there are 67 missing values expressed by the “?” symbol in Fig. 11. The criterion values support(τ) and accuracy(τ) are affected by the value assigned to “?” symbol [15, 24]. Of course, rules depend upon the missing values; so, rule generation from such table data sets [11, 12, 18, 21] is different from rule generation from DISs. We have dealt with this problem in RNIA.

We introduced the concept of derived DISs to NIS. If we replace each “?” symbol with a possible value, then we have one DIS. We termed such a DIS a derived DIS. An example of NIS and derived DISs is given in the twinned paper [23]. We employ the usual definition of a rule in DIS [13], and extended it to a certain rule and a possible rule in NIS below [15, 16]:

A rule in DIS is a rule in NIS, if τ satisfies support(τ) ≥ α and accuracy(τ) ≥ β for the given α and β. A certain rule in NIS is a certain rule, if τ is a rule in a derived DIS from NIS for the given α and β. A possible rule in NIS is an implication τ is a possible rule, if τ is a rule in at least one derived DIS from NIS for the given α and β.

Generally, it will be hard to generate certain rules and possible rules, if there are a huge number of possible worlds. In the Mammographic data set, the number of possible worlds exceeds 10^{100}. However, we have proved the following [16, 19] and described them in the twinned paper.

(i) There exists a derived DIS which causes both the support(τ) value and the accuracy(τ) value to be minimum. We term such values minsupp and minacc for τ.
(ii) There exists a derived DIS which causes both the support(τ) value and the accuracy(τ) value to be maximum. We term such values maxsupp and maxacc for τ.
(iii) There is a method, which does not depend upon the amount of possible worlds, to calculate minsupp, minacc, maxsupp, and maxacc for τ. The formulas are in the twinned paper.
(iv) An implication τ is a certain rule if and only if minsupp(τ) ≥ α and minacc(τ) ≥ β. An implication τ is a possible rule if and only if maxsupp(τ) ≥ α and maxacc(τ) ≥ β.
(v) The complexity of NIS-Apriori is more than twice of complexities (or linear-order time complexity) of the DIS-Apriori algorithm. Since the DIS-Apriori algorithm is an adjusted Apriori algorithm, the NIS-Apriori algorithm will be linear-order time complexity of the Apriori algorithm by Agrawal. Even though the NIS-Apriori algorithm can handle rules defined by possible worlds, which increase exponentially, the NIS-Apriori algorithm can escape from the exponential time-order complexity.

In the subsequent sections, we employ the above results and realise a rule generator from NISs. Furthermore, we realise some decision support procedures by using the obtained rules.

3.2 Rule generator in SQL for NISs

For rule generation from NISs, we realised procedures step1, step2, and step3 below:

$$\text{step1} \left( \text{dec\_attr}, \{OB\}, \alpha, \beta \right),$$
$$\text{step2} \left( \text{dec\_attr}, \text{dec\_attr\_value}, \{OB\}, \alpha, \beta \right),$$
$$\text{step3} \left( \text{dec\_attr}, \{OB\}, \alpha, \beta \right),$$

In step1, the DIS-Apriori algorithm with the minsupp and minacc values is called for certain rule generation, and then the DIS-Apriori algorithm with the maxsupp and maxacc values is called for possible rule generation. In step2 and step3, the similar procedure is executed. Fig. 12 shows the generated tables. Tables c, s store certain rules and tables p, s store possible rules. Fig. 13 shows certain rules in c1_rule and possible rules in p2_rule. In the
We say that an implication $A \land B \implies C$ is redundant to the implication $A \implies C$. After obtaining rules, we see them logical implications in propositional logic. If $A \implies C$ holds, $A \land B \implies C$ automatically holds, because $(A \land B \implies C) = (A \implies C) \lor (B \implies C)$. Thus, we see $(A \land B \implies C)$ is a rule, if $A \implies C$ is recognised as a rule. (If we consider large numbers of descriptors in the condition part, the constraint $\text{support} \geq \alpha$ may not be satisfied. So, actually such an implication may not be a rule.) We employ this redundancy concept, and generate rules with the minimal condition part (minimal complex) in [24]. Otherwise, there will be large numbers of redundant rules to one rule, like $A \land B \implies C$, $A \land D \implies C$, $A \land E \implies C$, ..., to a rule $A \implies C$. We are interested in rules with minimal condition part.

In Fig. 14, an implication $[\text{t, ear}, 1] \implies [\text{class}, 3]$ is a rule, and we automatically see (any descriptor) $A [\text{t, ear}, 1] \implies [\text{class}, 3]$ is a rule. In decision support, we need to pay attention to this redundancy.

4.2 Decision support environment for DISs: phase (ii)

For decision support in DISs, we realised procedures $sd\_rule1$, $sd\_rule2$, and $sd\_rule3$ below:

$$sd\_rule1(\text{dec}\_attrib, \text{attrib}1, \text{value}1), \quad sd\_rule2(\text{dec}\_attrib, \text{attrib}1, \text{value}1, \text{attrib}2, \text{value}2), \quad sd\_rule3(\text{dec}\_attrib, \text{attrib}1, \text{value}1, \text{attrib}2, \text{value}2, \text{attrib}3, \text{value}3).$$

These procedures search rules stored in tables $rule1$, $rule2$, and $rule3$. Let us consider Figs. 6 and 8. For such tables, we apply $sd\_rule1$ and $sd\_rule2$ in Fig. 14.

The first query is to ‘decide decision for $[\text{t, ear}, 1]$ by the obtained rules’. Since $\tau [\text{t, ear}, 1] \implies [\text{class}, 3]$ is stored in $rule1$, the decision $[\text{class}, 3]$ is obtained by $\tau$. The second query is to ‘decide decision for $[\text{spec}, 1] \land [\text{t, ear}, 1]$ by the obtained rules’. In this case, $\tau [\text{spec}, 1] \land [\text{t, ear}, 1] \implies [\text{class}, 3]$ is redundant to $\tau$, so, the decision by $\tau$ is obtained. In $rule2$, there is no rule with $[\text{spec}, 1] \land [\text{t, ear}, 1]$ as the condition part.

The role of the described procedures may not be important for small number of rules. However, if we employ lower $\alpha$ and lower $\beta$ values, we have large number of rules. To examine the constraints manually seems to be ineffective. In such case, the realised $sd\_rule1$, $sd\_rule2$, and $sd\_rule3$ become very important.

4.3 Decision support environment for NISs: phase (ii)

For decision support in NISs, we realised procedures $sn\_rule1$, $sn\_rule2$, and $sn\_rule3$ below:

$$sn\_rule1(\text{dec}\_attrib, \text{attrib}1, \text{value}1), \quad sn\_rule2(\text{dec}\_attrib, \text{attrib}1, \text{value}1, \text{attrib}2, \text{value}2), \quad sn\_rule3(\text{dec}\_attrib, \text{attrib}1, \text{value}1, \text{attrib}2, \text{value}2, \text{attrib}3, \text{value}3).$$

The procedures $sn\_rule1$, $sn\_rule2$, and $sn\_rule3$ are almost the same procedures of $sd\_rule1$, $sd\_rule2$, and $sd\_rule3$. Three procedures search rules stored in tables $c1\_rule$, $c2\_rule$, $c3\_rule$, $p1\_rule$, $p2\_rule$.
p2_rule, and p3_rule. Let us consider Figs. 12 and 13. For such tables, we apply sn_rule1 and sn_rule2 in Fig. 15. The first query is ‘to decide decision for [a5, p] by the obtained rules’. Since \( \pi(a5, p) \) by the obtained rules in c1_rule and p1_rule, the decision \([a16, 1]\) is obtained by using \( \pi \). We know the degree of \( \tau \) by \( \minsupp(\tau), \minacc(\tau), \maxsupp(\tau) \), and \( \maxacc(\tau) \). The second query is ‘to decide decision for \([a5, g] \land [a7, h] \) by the obtained rules’. In this case, rules with \([a5, g] \) and \([a7, h] \) are not stored in c1_rule nor p1_rule, and a rule with \([a5, g] \land [a7, h] \) is stored in p2_rule. Even though the support value may not be enough, we will probably conclude \([a16, 2]\).

Like this, the proposed and implemented procedures will be effective for decision support. The procedures search the obtained rules in tables; so, it takes less execution time. However, if the condition does not match the obtained rules, we have no information for the condition.

### 4.4 Decision support environment for DISs: phase (iii)

In order to recover the case that the condition does not match the obtained rules in DISs, we implemented the procedures srdf1, srdf2, and srdf3 below:

\[
\begin{align*}
\text{srdf1}(\text{dec_attrib}, \text{OB}, \text{attrib1}, \text{value1}), \\
\text{srdf2}(\text{dec_attrib}, \text{OB}, \text{attrib1}, \text{value1}, \text{attrib2}, \text{value2}), \\
\text{srdf3}(\text{dec_attrib}, \text{OB}, \text{attrib1}, \text{value1}, \text{attrib2}, \text{value2}, \text{attrib3}, \text{value3}).
\end{align*}
\]

Each procedure in the formulas (7) searches the rdf table instead of tables with the obtained rules. Therefore, it takes more execution time than that by sd_rule1, sd_rule2, and sd_rule3. However, we have all information about the specified condition.

In Fig. 16, the procedure sd_rule2 is executed at first, and we have no information on the condition \([\text{age}, 1] \land [\text{spec}, 1] \). Namely, any criterion value is not so characteristic, but if necessary we will probably have the decision \([\text{class}, 3]\).

### 4.5 Decision support environment for NISs: phase (iii)

Similarly to the previous subsection, we implemented the procedures snrdf1, snrdf2, and snrdf3. Each procedure in the formulas (8) searches the snrdf table instead of tables with the obtained rules. Therefore, it takes more execution time than that by sn_rule1, sn_rule2, and sn_rule3. However, we have all information about the implication with the specified condition.

\[
\begin{align*}
\text{snrdf1}(\text{dec_attrib}, \text{OB}, \text{attrib1}, \text{value1}), \\
\text{snrdf2}(\text{dec_attrib}, \text{OB}, \text{attrib1}, \text{value1}, \text{attrib2}, \text{value2}), \\
\text{snrdf3}(\text{dec_attrib}, \text{OB}, \text{attrib1}, \text{value1}, \text{attrib2}, \text{value2}, \text{attrib3}, \text{value3}).
\end{align*}
\]

In Fig. 17, the procedure sn_rule1 is executed at first, and we have no information on the condition \([\text{a1}, \text{a}] \). Namely, any rule
[a1, a] ⇒ [class, val] is not stored in c1_rule nor p1_rule. In this case, we execute snrdf procedure, and have the table snrdf. This is all information about the condition [a1, a]. Criterion values are not so characteristic, but we will probably have the decision [a16, 1].

Fig. 18 shows the relation on two implications in the table snrdf1. In this case, the following holds:

\[ \maxsupp([a1, a] \Rightarrow [a16, 2]) \leq \minsupp([a1, a] \Rightarrow [a16, 1]), \]
\[ \maxacc([a1, a] \Rightarrow [a16, 2]) \leq \minacc([a1, a] \Rightarrow [a16, 1]). \] (9)

Even though they do not satisfy the constraint of a rule, [a1, a] ⇒ [a16, 1] takes better criterion values in any possible case. Thus, we will have a decision [a16, 1], if we need to have one decision.

4.6 Decision support in DISs: A case of the phishing data set

This subsection shows an example on the Web Phishing data set in UCI machine learning repository [22]. This data set stores characteristics of web sites. This data set consists of 1353 objects, 9 condition attributes, and one decision attribute Result with three attribute values 1 (Legitimate), 0 (Suspicious), −1 (Phishy). The apr2v2 procedure generated 21 rules for \( \text{support} \geq 0.1 \) and \( \text{accuracy} \geq 0.7 \). It took about 400 s. Fig. 19 shows decision support by using the obtained results.

In Fig. 19, the table srdf2 shows all cases for the condition URL of Anchor, 1] ∧ [web_traffic, −1]. The sum of accuracy values is 1. From srdf2, we will have the decision [Result, −1] (Phishy). In sd_rule2, this implication is stored as a rule.

4.7 Decision support in NISs: A case of the congressional voting data set

This subsection shows an example on the Congressional Voting data set. This data set stores congressional voting in US congress. This data set consists of 435 objects, 16 condition attributes, and one decision attribute whose attribute values are rep(ublican) political party and dem(ocrat) political party.

The step1 procedure generated 18 certain rules in c1_rule and 21 possible rules in c1_rule for \( \text{support} \geq 0.3 \) and \( \text{accuracy} \geq 0.7 \). It took 18.71 s. Similarly, step2 did two certain rules in c2_rule and eight possible rules in p2_rule. It took 17.43 s. No rule was generated by step 3. The certain rules satisfy support(\( \tau \)) ≥ 0.3 and accuracy(\( \tau \)) ≥ 0.7 for each of more than 10^30 derived DISs. Especially, two certain rules [t5, u] ⇒ [t1, dem] and [t5, y] ⇒ [t1, rep] are very strong. If we have a person’s answer to the attribute t5, we will easily conclude his supporting party by using two tables in Fig. 20.

4.8 Discussion about the decision support functionality

We applied the implemented procedures to the obtained tables, and we had one curious case in Fig. 21. In Fig. 21, both [a12, f] and [a4, u] conclude [a16, 1], but [a12, f] ∧ [a4, u] concludes [a16, 2]. This seems to confuse us.

The reason was the definition of rule. The constraint was support(\( \tau \)) ≥ 0.2 and accuracy(\( \tau \)) ≥ 0.5, and the accuracy value seems to be lower. In possible rule generation, we calculate the maxacc(\( \tau \)) value in the most suitable case for \( \tau \). Therefore, we have two inconsistent possible rules in Fig. 22, i.e.
\[ [a_{12}, f] \land [a_{4}, u] \Rightarrow [a_{16}, 1] \ (\text{maxsupp} = 0.206, \ \text{maxacc} = 0.504) \]
and \[ [a_{12}, f] \land [a_{4}, u] \Rightarrow [a_{16}, 2] \ (\text{maxsupp} = 0.209, \ \text{maxacc} = 0.507) \]. This is not wrong, and such case may occur based on the definition of possible rules. Actually, we changed the condition to \( \text{accuracy}(\tau) \geq 0.7 \), and such curious cases were solved.

In decision support, the application of phases (ii) and (iii) in Figs. 1 and 2 will be effective. In the RNIA framework, rules and decisions are evaluated by the criterion values \( \text{support} \) and \( \text{accuracy} \). The proposed and implemented procedures provide all information on \( \text{support}(\tau) \) and \( \text{accuracy}(\tau) \). However, if there is less difference on the \( \text{support} \) and \( \text{accuracy} \) values like in Figs. 21 and 22, it may be difficult to have one decision. We may need other criterion values in such cases.

5 Concluding remarks

This paper clarified rule-based decision support in RNIA, and reported its prototype system in SQL. The definition of the certain rules and the possible rules seems to be natural, however there is less software tool for handling them, because the rules are defined by all derived DISs whose number may exceed \( 10^{10} \). Without effective property, it will be hard to obtain rules. The NIS-Apriori algorithm affords a solution to this problem, and we implemented the prototype termed NIS-Apriori in SQL. This algorithm takes the core part for handling the uncertainty, and we applied it to decision support environment. Rule generation and decision support based on NISs have not been investigated previously; therefore, each research related to the NIS-Apriori algorithm and NISs will be novel research.

In the application of the obtained rules, we proposed some procedures and implemented them in SQL. If we have large number of rules, it will be hard to examine phase (ii) in Figs. 1 and 2 manually. We gave a solution to this problem. Actually, to search tables with rules is less time-consuming, but we cannot have one decision for any condition always. Probably, in most case, we have no information. Thus, we realised procedures to search either \( \text{rdfs} \) or \( \text{ndrs} \) tables. The implemented procedures will be effective for decision support.

Fig. 20 If his response is ‘no’ to the attribute \( a_5 \), he will support democrat party. If his response is ‘yes’ to the attribute \( a_5 \), he will support republic party.

Fig. 21 Curious case in the Credit Approval data set

Fig. 22 Application of \( \text{snrdfs} \) and \( \text{snrdfs} \) to the Credit Approval data set
This paper described the actual executions whose theoretical aspect was presented in the twinned previous paper entitled ‘Rough set-based rule generation and Apriori-based rule generation from table data sets: A survey and a combination’.

6 References

[1] Elsevier, Scopus system. Available at https://www.scopus.com, accessed December 2018
[2] Dubois, D., Prade, H.: ‘Rough fuzzy sets and fuzzy rough sets’, Int. J. Gen. Syst., 1990, 17, (2–3), pp. 191–209
[3] Jensen, R., Shen, Q.: ‘Fuzzy rough sets assisted attribute selection’, IEEE Trans. Fuzzy Syst., 2007, 15, (1), pp. 73–89
[4] Minutolo, A., Esposito, M., De Pietro, G.: ‘A fuzzy framework for encoding uncertainty in clinical decision-making’, Knowl-Based Syst., 2016, 98, pp. 95–116
[5] Shen, K.Y., Sakai, H., Tzeng, G.H.: ‘Comparing two novel hybrid MRDM approaches to consumer credit scoring under uncertainty and fuzzy judgments’, Int. J. Fuzzy Syst., 2019, 21, (1), pp. 194–212
[6] Skowron, A., Rauszer, C.: ‘The discernibility matrices and functions in information systems’, in Slowinski, R. (Ed.): ‘Intelligent decision support – handbook of advances and applications of the rough set theory’ (Kluwer Academic Publishers, the Netherlands, 1992, 1st edn.), pp. 331–362
[7] Zadeh, L.: ‘Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic’, Fuzzy Sets Syst., 1997, 90, (2), pp. 111–127
[8] Zawiski, V., Papargyriou, E., Regner, P.: ‘Bayesian network construction using a fuzzy rule based approach for medical decision support’, Expert Syst., 2016, 32, (3), pp. 344–369
[9] Kadrziński, M., Słowiński, R., Slezak, M.: ‘Dominance-based rough set approach to multiple criteria ranking with sorting-specific preference information’, Studies in Comput. Intell., 2016, 606, pp. 55–171
[10] Komorowski, J., Pawlak, Z., Polkowski, L., et al.: ‘Rough sets: a tutorial’, in Pal, S.K., Skowron, A. (Eds.): ‘Fuzzy rough hybridization: a new method for decision making’ (Springer, German, 1999, 1st edn.), pp. 3–98
[11] Nakata, M., Sakai, H.: ‘Twofold rough approximations under incomplete information’, Int. J. Gen. Syst., 2013, 42, (6), pp. 546–571
[12] Pawlak, Z.: ‘Rough sets: theoretical aspects of reasoning about data’ (Kluwer Academic Publishers, Dordrecht, The Netherlands, 1991, 1st edn.)