An Attribute-Value Block Based Method of Acquiring Minimum Rule Sets: A Granulation Method to Construct Classifier

Zuqiang Meng(✉) and Qiuling Gan

College of Computer, Electronics and Information, Guangxi University, Nanning 530004, Guangxi, China
zqmeng@126.com

Abstract. Decision rule acquisition is one of the important topics in rough set theory and is drawing more and more attention. In this paper, decision logic language and attribute-value block technique are introduced first. And then realization methods of rule reduction and rule set minimum are relatively systematically studied by using attribute-value block technique, and as a result effective algorithms of reducing decision rules and minimizing rule sets are proposed, which, together with related attribute reduction algorithm, constitute an effective granulation method to acquire minimum rule sets, which is a kind classifier and can be used for class prediction. At last, related experiments are conducted to demonstrate that the proposed methods are effective and feasible.

Keywords: Rule acquisition · Attribute-value blocks · Decision rule set · Classifier

1 Introduction

Rough set theory [1], as a powerful mathematical tool to deal with insufficient, incomplete or vague information, has been widely used in many fields. In rough set theory, the study of attribute reduction seems to attract more attention than that of rule acquisition. But in recent years there have been more and more studies involving the decision rule acquisition. Papers [2, 3] gave discernibility matrix or the discernibility function-based methods to acquire decision rules. These methods are able to acquire all minimum rule sets for a given decision system theoretically, but they usually would pay both huge time cost and huge space cost, which extremely narrow their applications in real life. In addition, paper [4] discussed the problem of producing a set of certain and possible rules from incomplete data sets based on rough sets and gave corresponding rule learning algorithm. Paper [5] discussed optimal certain rules and optimal association rules, and proposed two quantitative measures, random certainty factor and random coverage factor, to explain relationships between the condition and decision parts of a rule in incomplete decision systems. Paper [6] also discussed the rule acquisition in incomplete decision contexts. This paper presented the notion of an approximate decision rule, and then proposed an approach for extracting non-redundant approximate decision rules from an incomplete decision context. But the proposed method is also based on discernibility matrix and discernibility function, which determines that it is relatively difficult to acquire decision rules from large data sets.

Attribute-value block technique is an important tool to analyze data sets [7, 8]. Actually, it is a granulation method to deal with data. Our paper will use the attribute-value block technique and other related techniques to systematically study realization methods of rule reduction and rule set minimum, and propose effective algorithms of reducing decision rules and minimizing decision rule sets. These algorithms, together with related attribute reduction algorithm, constitute an effective solution to the acquisition of minimum rule sets, which is a kind classifier and can be used for class prediction.

The rest of the paper is organized as follows. In Sect. 2, we review some basic notions linked to decision systems. Section 3 introduces the concept of minimum rule sets. Section 4 gives specific algorithms for rule reduction and rule set minimum based on attribute-value blocks. In Sect. 5, some experiments are conducted to verify the effectiveness of the proposed methods. Section 6 concludes this paper.

2 Preliminaries

In this section, we first review some basic notions, such as attribute-value blocks, decision rule sets, which are prepared for acquiring minimum rule sets in next sections.

2.1 Decision Systems and Relative Reducts

A decision system (DS) can be expressed as the following 4-tuple: $DS = (U, A = C \cup D, V = \bigcup_{a \in A} V_a, \{f_a\})$, where U is a finite nonempty set of objects; C and D are condition attribute set and decision attribute set, respectively, and $C \cap D = \emptyset$; V_a is a value domain of attribute a; f_a: $U \rightarrow V$ is an information function from U to V, which maps an object in U to a value in V_a.

For simplicity, $(U, A = C \cup D, V = \bigcup_{a \in A} V_a, \{f_a\})$ is expressed as $(U, C \cup D)$ if V and f_a are understood. Without loss of generality, we suppose D is supposed to be composed of only one attribute.

For any $B \subseteq C$, let $U/B = \{[x]_B \mid x \in U\}$, where $[x]_B = \{y \in U \mid f_a(y) = f_a(x)$ for any $a \in B\}$, which is known as equivalence class. For any subset $X \subseteq U$, the lower approximation $\underline{BX}$ and the upper approximation $\overline{BX}$ of X with respect to B are defined by: $\underline{BX} = \{x \in U \mid [x]_B \subseteq X\}$, $\overline{BX} = \{x \in U \mid [x]_B \cap X \neq \phi\}$. And then the concepts of positive region $POS_B(X)$, boundary region $BND_B(X)$ and negative region $NEG_B(X)$ of X are defined as: $POS_B(X) = \underline{BX}$, $BND_B(X) = \overline{BX} - \underline{BX}$, $NEG_B(X) = U - \overline{BX}$.

Suppose that $U/D = \{[x]_D \mid x \in U\} = \{D_1, D_2, \ldots, D_m\}$, where $m = |U/D|$, D_i is a decision class, $i \in \{1, 2, \ldots, m\}$. Then for any $B \subseteq C$, the concepts of positive region $POS_B(D)$, boundary region $BND_B(D)$ and negative region $NEG_B(D)$ of a decision system $(U, C \cup D)$ can be defined as follows:

$$POS_B(D) = POS_B(D_1) \cup POS_B(D_2) \cup \ldots \cup POS_B(D_m),$$
$$BND_B(D) = BND_B(D_1) \cup BND_B(D_2) \cup \ldots \cup BND_B(D_m),$$
$$NEG_B(D) = U - POS_B(D) \cup BND_B(D).$$

With the positive region, the concept of reducts can be defined as follows: given a decision system $(U, C \cup D)$ and $B \subseteq C$, B is a **relative reduct** of C with respect to D if the following conditions are satisfied: (1) $POS_B(D) = POS_C(D)$, and (2) for any $a \in B, POS_{B-\{a\}}(D) \neq POS_B(D)$.

2.2 Decision Logic and Attribute-Value Blocks

Decision rules are in fact related formulae in decision logic. In rough set theory, a decision logic language depends on a specific information system, while a decision system $(U, C \cup D)$ can be regarded as being composed of two information systems: (U, C) and (U, D). Therefore, there are two corresponding decision logic languages, while attribute-value blocks just act as a bridge between the two languages. For the sake of simplicity, let $IS(B) = (U, B, V = \bigcup_{a \in A} V_a, \{f_a\})$ is an information system with respect to B, where $B \subseteq C$or $B \subseteq D$. Then a decision logic language $DL(B)$ is defined as a system being composed of the following formulae [3]:

(1) (a, v) is an atomic formula, where $a \in B, v \in V_a$;
(2) an atomic formula is a formula in $DL(B)$;
(3) if φ is a formula, then $\sim\varphi$ is also a formula in $DL(B)$;
(4) if both φ and ψ are formulae, then $\varphi\vee\psi$, $\varphi\wedge\psi$, $\varphi \rightarrow \psi$, $\varphi \equiv \psi$ are all formulae;
(5) only the formulae obtained according to the above Steps (1) to (4) are formulae in $DL(B)$.

The atomic formula (a, v) is also called **attribute-value pair** [7]. If φ is a simple conjunction, which consists of only atomic formulae and connectives $\wedge$, then φ is called a **basic formula**.

For any $x \in U$, the relationship between x and formulae in $DL(B)$ is defined as following:

(1) $x| = (a, v)$ iff $f_a(x) = v$;
(2) $x| = \sim\varphi$ iff not $x| = \varphi$;
(3) $x| = \varphi \wedge \psi$ iff $x| = \varphi$ and $x| = \psi$;
(4) $x| = \varphi \vee \psi$ iff $x| = \varphi$ or $x| = \psi$;
(5) $x| = \varphi \rightarrow \psi$ iff $x| = \sim\varphi \vee \psi$;
(6) $x| = \varphi \equiv \psi$ iff $x| = \varphi \rightarrow \psi$ and $x| = \psi \rightarrow \varphi$.

For formula φ, if $x| = \varphi$, then we say that the object x satisfies formula φ. Let $[\varphi] = \{x \in U | x| = \varphi\}$, which is the set of all those objects that satisfy formula φ. Obviously, formula φ consists of several attribute-value pairs by using connectives. Therefore, $[\varphi]$ is so-called an **attribute-value block** and φ is called the (attribute-value pair) **formula** of the block. For $DL(C)$ and $DL(D)$, they are distinct decision logic languages and have no formulae in common. However, through attribute-value blocks, an association between $DL(C)$ and $DL(D)$ can be established. For example, suppose $\varphi \in DL(C)$ and $\psi \in DL(D)$ and obviously φ and ψ are two different formulae; but if $[\varphi] \subseteq [\psi]$, we can obtain a decision rule $\varphi \to \psi$. Therefore, attribute-value blocks play an important role in acquiring decision rules, especially in acquiring certainty rules.

3 Minimum Rule Sets

Suppose that $\varphi \in DL(C)$ and $\psi \in DL(D)$. Implication form $\varphi \to \psi$ is said to be a (**decision**) **rule** in decision system $(U, C \cup D)$. If both φ and ψ are basic formula, then $\varphi \to \psi$ is called **basic decision rule**. A decision rule is not necessarily useful unless it satisfies some given indices. Below we introduce these indices.

A decision rule usually has two important measuring indices, confidence and support, which are defined as: $conf(\varphi \to \psi) = |[\varphi] \cap [\psi]|/|[\varphi]|, sup(\varphi \to \psi) = |[\varphi] \cap [\psi]|/|U|$, where $conf(\varphi \to \psi)$ and $sup(\varphi \to \psi)$ are **confidence** and **support** of decision rule $\varphi \to \psi$, respectively.

For decision system $DS = (U, C \cup D)$, if rule $\varphi \to \psi$ is true in $DL(C \cup D)$, i.e., for any $x \in Ux| = \varphi \to \psi$, then rule $\varphi \to \psi$ is said to be **consistent** in *DS*, denoted by $| =_{DS} \varphi \to \psi$; if there exists at least object $x \in U$ such that $x| = \varphi \wedge \psi$, then rule $\varphi \to \psi$ is said to be **satisfiable** in *DS*. Consistency and satisfiability are the basic properties that must be satisfied by decision rules.

For object $x \in U$ and decision rule r: $\varphi \to \psi$, if $x \,| = r$, then it is said that rule r **covers** object x, and let **coverage**$(r) = \{x \in U|x| = r\}$, which is the set of all objects that are **covered** by rule r; for two rules, r_1 and r_2, if coverage$(r_1) \subseteq$ coverage(r_2), then it is said that r_2 **functionally covers** r_1, denoted by $r_1 \leq r_2$. Obviously, if there exist such two rules, then rule r_1 is redundant and should be deleted, or in other words, those rules that are functionally covered by other rules should be removed out from rule sets.

In addition, for a rule $\varphi \to \psi$, we say that $\varphi \to \psi$ is **reduced** if $[\varphi] \subseteq [\psi]$ does not hold any more when any attribute-value pair is removed from φ. And this is just known as rule reduction, which will be introduced in next section.

A decision rule set $\wp$ is said to be **minimal** if it satisfies the following properties [3]: (1) any rule in $\wp$ should be consistent; (2) any rule in $\wp$ should be satisfiable; (3) any rule in $\wp$ should be reduced; (4) for any two rules r_1, $r_2 \in \wp$, neither $r_1 \leq r_2$ nor $r_2 \leq r_1$.

In order to obtain a minimum rule set from a given data set, it is required to complete three steps: attribute reduction, rule reduction and rule set minimum. This paper does not introduce attribute reduction methods any more, and we try to propose new methods for rule reduction and for rule set minimum in next sections.

4 Methods of Acquiring Decision Rules

4.1 Rule Reduction

Rule reduction is to keep the minimal attribute-value pairs in a rule such that the rule is still consistent and satisfiable by removing redundant attributes from the rule. For the convenience of discussion, we let $r(x)$ denote a decision rule that is generated with object x, and introduce the following definitions and properties.

Definition 1. For decision system $DS = (U, C \cup D), B = \{a_1, a_2, \ldots, a_m\} \subseteq C$ and $x \in U$, let $\boldsymbol{pairs}(\boldsymbol{x}, \boldsymbol{B}) = (a_1, f_{a_1}(x)) \wedge (a_2, f_{a_2}(x)) \wedge \ldots \wedge (a_m, f_{a_m}(x))$ and let $\boldsymbol{block}(\boldsymbol{x}, \boldsymbol{B}) = [pairs(x, B)] = [(a_1, f_{a_1}(x)) \wedge (a_2, f_{a_2}(x)) \wedge \ldots \wedge (a_m, f_{a_m}(x))]$, and the number m is called the lengths of $pairs(x, B)$ and $block(x, B)$, denoted by $|pairs(x, B)|$ and $|block(x, B)|$, respectively.

Property 1. Suppose $B_1, B_2 \subseteq C$ with $B_1 \subseteq B_2$, then $block(x, B_2) \subseteq block(x, B_1)$.

The proof of Property 1 is straightforward. According to this property, for an attribute subset B, $block(x, B)$ increases with removing attributes from B, but with the prerequisite that $block(x, B)$ does not "exceed" the decision class $[x]_D$, to which x belongs. Therefore, how to judge whether $block(x, B)$ is still contained in $[x]_D$ or not is crucial for rule reduction.

Property 2. For decision system $DS = (U, C \cup D)$ and $B \subseteq C, block(x, B) \subseteq [x]_D$ $(= block(x, D))$ if and only if $f_d(y) = f_d(x)$ for all $y \in block(x, B)$.

The proof of Property 2 is also straightforward. This property shows that the problem of judging whether $block(x, B)$ is contained in $[x]_D$ becomes that of judging whether $f_d(y) = f_d(x)$ for all $y \in block(x, B)$. Evidently, the latter is much easier than the former. Thus, we give the following algorithm for reducing a decision rule.

```
Algorithm 1: an algorithm for reducing a decision rule
Input: decision system ($U$, $C \cup D$) and object $x \in U$, where $C = \{a_1, a_2, ..., a_n\}$, $n = |C|$, $D = \{d\}$
Output: reduced decision rule $r(x)$
Begin
  Step 1.   Let $B = C$;
  Step 2.   For $i = 1$ to $|C|$ do
  Step 3.       Let $B = B - \{a_i\}$;
  Step 4.       Compute $block(x, B)$;
  Step 5.       For each $y \in block(x, B)$ do
  Step 6.           If $f_d(y) \neq f_d(x)$ then { Let $B = B \cup \{a_i\}$; break; }
  Step 7.       Let $\varphi = pairs(x, B)$ and $\psi = (d, f_d(x))$;
  Step 8.   Let $r(x) = \varphi \rightarrow \psi$;
  Step 9.   return $r(x)$;
End.
```

The time-consuming step in this algorithm is to compute $block(x, B)$, whose comparison number is $|U||B|$. Therefore, the complexity of this algorithm is $O(|U||C|^2)$ in the worst case. According to Algorithm 1, it is guaranteed at any time that $block(x, B) \subseteq [x]_D = block(x, D)$, so the confidence of rule $r(x)$ is always equal to 1.

4.2 Minimum of Decision Rule Sets

Using Algorithm 1, each object in U can be used to generate a rule. This means that after reducing rules, there are still $|U|$ rules left. Obviously, there must be many rules that are covered by other rules, and hereby we need to delete those rules which are covered by other rules.

For decision system $(U, C \cup D)$, after using Algorithm 1 to reduce each object $x \in U$, all generated rules $r(x)$ constitute a rule set, denoted by RS, i.e., $RS = \{r(x)\,|x \in U\}$. Obviously, $|RS| = |U|$. Our purpose in this section is to delete those rules which are covered by other rules, or in other words, to minimize RS such that each of the remaining rules is consistent, satisfiable, reduced, and is not covered by other rules.

Suppose $V_d = \{v_1, v_2, \ldots, v_t\}$. We use decision attribute d to partition U into t attribute-value blocks (equivalence classes): $[(d, v_1)], [(d, v_2)], \ldots, [(d, v_t)]$. Let $U_{v_i} = [(d, v_i)]$, and thus $\bigcup_{i\in\{1,2,\ldots,t\}} U_{v_i} = U$ and $U_{v_i} \cap U_{v_j} = \emptyset$, where $i \neq j, i, j \in \{1, 2, \ldots, t\}$. Accordingly, let $RS_{v_i} = \{r(x)\,|\,x \in U_{v_i}\}$, where $i \in \{1, 2, \ldots, t\}$. Obviously, $\{RS_{v_i} \mid i \in \{1,2,\ldots,t\}\}$ is a partition of RS. According to Algorithm 1, for any $r' \in RS_{v_i}$ and $r'' \in RS_{v_j}$, where $i \neq j$, neither $r' \leq r''$ nor $r'' \leq r'$, because coverage$(r') \subseteq U_{v_i}$ while coverage$(r'') \subseteq U_{v_j}$ and then coverage$(r') \cap$ coverage$(r'') = \emptyset$. This means that a rule in RS_{v_i} does not functionally covers any rule in RS_{v_j}. Thus, we can independently minimize each RS_{v_i}, and the union of all the generated rule subsets is the final minimum rule set that we want.

Let independently consider RS_{v_i}, where $i \in \{1, 2, \ldots, t\}$. For $r(x) \in RS_{v_i}$, if there exists $r(y) \in RS_{v_i}$ such that $r(x) \leq r(y)(r(y)$ functionally covers $r(x))$, where $x \neq y$, then $r(x)$ should be removed from RS_{v_i}, otherwise it should not. Suppose after removing, the set of all remaining rules in RS_{v_i} is denoted by RS'_{v_i}, and thus we can give an algorithm for minimizing RS_{v_i}, which is described as follows.

Algorithm 2: an algorithm for minimizing RS_{v_i}

Input: $U_{v_i} = \{x_1, x_2, ..., x_q\}$ and $RS_{v_i} = \{r(x_1), r(x_2), ..., r(x_q)\}$

Output: RS'_{v_i}

```
Begin
  Step 1.    Let $\wp = RS_{v_i}$;
  Step 2.    For $j = 1$ to $q$ do                    // $q = |RS_{v_i}|$
  Step 3.        Let $\wp = \wp - \{r(x_j)\}$;
  Step 4.        Let flag = 1;
  Step 5.        For each $r \in \wp$ do
  Step 6.            If $x_j \in$ coverage($r$) then { flag = 0; break; }
  Step 7.            If flag = 1 then $\wp = \wp \cup \{r(x_j)\}$;
  Step 8.        Let $RS'_{v_i} = \wp$;
  Step 9.    Return $RS'_{v_i}$;
End.
```

In Algorithm 2, judging if $x_j \in \text{coverage}(r)$ takes at most $|C|$ comparison times. But because all rules in RS_{v_i} have been reduced by Algorithm 1, the comparison number should be much smaller than $|C|$. Therefore, the complexity of Algorithm 2 is $O(q^2 \cdot |C|) = O(|U_{v_i}|^2 \cdot |C|)$ in the worst case.

4.3 An Algorithm for Acquiring Minimum Rule Sets

Using the above proposed algorithms and related attribute reduction algorithms, we now can give an entire algorithm for acquiring a minimum rule set from a given data set. The algorithm is described as follows.

```
Algorithm 3: an algorithm for acquiring a minimum rule set from a data set
Input: decision system $DS = (U, C \cup D)$
Output: a minimum rule set, $minRS$
Begin
  Step 1.  Use an attribute reduction algorithm to find a reduct of $DS$, and
           suppose the reduct is $R$;
  Step 2.  Compute $U/R$, and then select one object in each equivalence class
           in $U/R$ to constitute a new decision system $(U', R \cup D)$;
  Step 3.  Reduce each object (rule) in $(U', R \cup D)$ using Algorithm 1, and
           suppose the obtained rule set is denoted by $RS$;
  Step 4.  Use decision attribute set $D$ to partition $U'$ into several decision
           classes: $U'_{v_1}, U'_{v_2}, ..., U'_{v_t}$, and then let $RS_{v_i} = \{r(x) \mid x \in U'_{v_i}\}$,
           where $i \in \{1,2,...,t\}$;
  Step 5.  In turn or in parallel minimize $RS_{v_1}, RS_{v_2}, ..., RS_{v_t}$ using Algorithm
           2, and suppose corresponding results are $RS'_{v_1}, RS'_{v_2}, ..., RS'_{v_t}$;
  Step 6.  Let $minRS = RS'_{v_1} \cup RS'_{v_2} \cup ... \cup RS'_{v_t}$;
  Step 7.  Return $minRS$;
End.
```

In Algorithm 3, there are three steps used to “evaporating” redundant data: Steps 2, 3, 5. These steps also determine the complexity of the entire algorithm. Actually, the newly generated decision system $(U', R \cup D)$ in Step 2 is completely determined by Step 1, which is attribute reduction and has the complexity of about $O(|C|^2|U|^2)$. The complexity of Step 3 is $O(|U'|^2|C|^2)$ in the worst case. Step 5’s complexity is $O(|U'_{v_1}|^2 \cdot |C|) + O(|U'_{v_2}|^2 \cdot |C|) + \ldots + O(|U'_{v_t}|^2 \cdot |C|)$. Because this step can be performed in parallel, so it can be more efficient under parallel environment. Generally, after attribute reduction, the size of a data set would greatly decrease, i.e., $|U'| << |U|$. Therefore, computation time of Algorithm 3 is mainly determined by Step 1, so it has the complexity of $O(|C|^2|U|^2)$ in most cases.

5 Experiment Analysis

This section aims to verify the effectiveness of the proposed methods through experiments. There are four UCI data sets (http://archive.ics.uci.edu/ml/datasets.html) used in our experiments, and they are outlined in Table 1. For missing values, they were replaced with the most frequently occurring value on the corresponding attribute.

We executed Algorithm 3 on the four data sets to obtain minimum rule sets. Suppose that the set of finally obtained decision rules on each data set is denoted by *minRS*. The indices that we are interesting in and their meanings are as follows.

- Number of rules: $|minRS|$, i.e., the number of decision rules in *minRS*
- Average value of support: $\frac{1}{|minRS|}\sum_{r \in minRS} sup(r)$, and $\mathrm{minValue} = \min_{r \in minRS}\{sup(r)\}$, $\mathrm{maxValue} = \max_{r \in minRS}\{sup(r)\}$
- Average value of confidence: $\frac{1}{|minRS|}\sum_{r \in minRS} conf(r)$
- Evaporation ratio: the ratio of removed items (attribute values) to all items (all attribute values)
- Running time: the running time of Algorithm 3, which includes attribute reduction, rule reduction and minimum of decision rule sets, and this index is measured in seconds.

The experimental results on the four data sets are shown in Table 2.

Table 1. Description of the four data sets.

No.	Data sets	Abbreviation	$\|U\|$	$\|C\|$	$\|V_d\|$
1	Dermatology database	Dermatology	366	34	6
2	Tic-Tac-Toe endgame database	Tic-Tac-Toe	958	9	2
3	Mushroom database	Mushroom	8124	22	2
4	Nursery database	Nursery	12960	8	5

Table 2. Experimental results on the four data sets

Data set	Number of rules	Average value of support (minValue, maxValue)	Average value of confidence	Evaporation ratio	Running time (Sec.)
Dermatology	72	0.0146 (0.0027, 0.1257)	1	0.9794	0.14
Tic-Tac-Toe	176	0.0066 (0.0010, 0.0940)	1	0.9072	0.16
Mushroom	17	0.0689 (0.0010, 0.2166)	1	0.9998	2.37
Nursery	305	0.0031 (0.00008, 0.3333)	1	0.9831	31.34

From Table 2, it can be found that the obtained rule sets on the four data sets all have very high evaporation ratio, and each rule in these rule sets has certain support. Specially, there are averagely 0.0689*8124 = 560 objects supporting each rule in the rule set obtained on Mushroom. This shows that these rule sets have relatively strong generalization ability. Furthermore, the running time of Algorithm 3 on each data set is not long and hereby can be accepted by users. In addition, Algorithm 1 can guarantee at any time that $block(x, B) \subseteq [x]_D = block(x, D)$ for all $x \in U$, so the confidence of each rule is always equal to 1, or in other words all the obtained decision rules are deterministic. All these results demonstrate Algorithm 3 is effective and has better application value.

6 Conclusion

Acquiring decision rules from data sets is an important task in rough set theory. This paper conducted our study through the following three aspects so as to provide an effective granulation method to acquire minimum rule sets. Firstly, we introduced decision logic language and attribute-value block technique. Secondly, we used attribute-value block technique to study how to reduce rules and to minimize rule sets, and then proposed effective algorithms for rule reduction and rule set minimum. Thus, together with related attribute reduction algorithm, the proposed granulation method constituted an effective solution to the acquisition of minimum rule sets, which is a kind classifier and can be used for class prediction. Thirdly, we conducted a series of experiments to show that our methods are effective and feasible.

Acknowledgements. This work is supported by the National Natural Science Foundation of China (No. 61363027), the Guangxi Natural Science Foundation (No. 2015GXNSFAA139292).

References

1. Pawlak, Z.: Rough set. Int. J. Comput. Inf. Sci. **11**(5), 341–356 (1982)
2. Guan, Y.Y., Wang, H.K., Wang, Y., Yang, F.: Attribute reduction and optimal decision rules acquisition for continuous valued information systems. Inf. Sci. **179**(17), 2974–2984 (2009)
3. Meng, Z., Jiang, L., Chang, H., Zhang, Y.: A heuristic approach to acquisition of minimum decision rule sets in decision systems. In: Shi, Z., Wu, Z., Leake, D., Sattler, U. (eds.) IIP VII. IFIP AICT, vol. 432, pp. 187–196. Springer, Heidelberg (2014)
4. Hong, T.P., Tseng, L.H., Wang, S.L.: Learning rules from incomplete training examples by rough sets. Expert Syst. Appl. **22**(4), 285–293 (2002)
5. Leung, Y., Wu, W.Z., Zhang, W.X.: Knowledge acquisition in incomplete information systems: a rough set approach. Eur. J. Oper. Res. **168**(1), 164–180 (2006)
6. Li, J.H., Mei, C.L., Lv, Y.J.: Incomplete decision contexts: approximate concept construction, rule acquisition and knowledge reduction. Int. J. Approximate Reasoning **54**(1), 149–165 (2013)
7. Grzymala-Busse, J.W., Clark, P.G., Kuehnhausen, M.: Generalized probabilistic approximations of incomplete data. Int. J. Approximate Reasoning **55**(1), 180–196 (2014)
8. Patrick, G.C., Grzymala-Busse, J.W.: Mining incomplete data with attribute-concept values and “do not care” conditions. In: IEEE International Conference on Big Data. IEEE (2015)