A Fast Granular Method for Classifying Incomplete Inconsistent Data

Zuqiang Meng(✉) and Hongli Li

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

Abstract. Today extracting knowledge from "inferior quality" data that is characterized by incompleteness and inconsistency is an unavoidable and challenging topic in the field of data mining. In this paper, we propose a fast granular method to classify incomplete inconsistent data using attribute-value block technique. Firstly, a granulation model is constructed to provide a foundation for efficient computation. Secondly, an algorithm of acquiring classification rules is proposed and then an algorithm of minimizing rule sets is proposed, and with these proposed algorithms, a classification algorithm is designed to construct a rule-based classifier. Finally, we use the experiment results to illustrate the effectiveness of the proposed algorithms.

Keywords: Attribute-value blocks · Data classification · Classifier · Incomplete inconsistent data · Granulation model

1 Introduction

Data classification is an important task in the field of data mining. Related classification methods and techniques are increasingly extensively studied and some of them have been successfully used to solve practical problems [1–3]. In the era of big data, the volume and variety of data is growing and growing. Big data shows features of not only large volume but also "inferior quality". "Inferior quality" of data embodies in many aspects, two of which are incompleteness and inconsistency. Data's incompleteness means that there are missing values in data while the inconsistency refers to that data contains conflicting descriptions. There are many reasons that cause the incompleteness and inconsistency, such as objective and subjective factors, noisy data, variety of data. In fact, problems caused by the incompleteness and inconsistency are unavoidable when extracting knowledge from big data [4]. The incompleteness usually enhances the degree of inconsistency. Actually the problems of incompleteness and inconsistency are interwoven and can not totally separated, which makes the problem of knowledge extraction more complicated. Therefore, it is difficult and meaningful to solve classification problems oriented to "inferior quality" data that is characterized mainly by incompleteness and inconsistency.

This work is supported by the National Natural Science Foundation of China (No. 61363027, 61762009), the Guangxi Natural Science Foundation (No. 2015GXNSFAA139292).

Z. Shi et al. (Eds.): ICIS 2017, IFIP AICT 510, pp. 153–163, 2017.
DOI: 10.1007/978-3-319-68121-4_16

There are many ways to handle missing values. Stefanowski [5]. distinguished two different semantics for missing values: the "absent value" semantics and the "missing value" semantics. Grzymala-Busse [6]. further divided such missing values into three categories according to their comparison range: "do not care" conditions, restricted "do not care" conditions, and attribute-concept values. In 2009 we utilized sorting technique to design a fast approach to compute tolerance classes [7]. But this approach has the problem of data fragmentation when the degree of missing values increases to a certain level. Recently, we presented a new method called index method to quickly compute attribute-value blocks [8], which can completely eliminate data fragmentations. In this paper, we studied on a fast classification problem of incomplete inconsistent data by using and improving such index methods.

2 Preliminaries

2.1 Incomplete Decision Systems (IDSs) and Attribute-Value Blocks

A decision system that contains missing values is called an incomplete decision system (IDS), which can be described as 4-tuple: $IDS = (U,\ A = C \cup D,\ V = \bigcup_{a \in A} V_a,\ \{f_a\})_{a \in A}$, where U is a finite nonempty set of objects, indicating a given universe; both C and D are finite nonempty set of attributes (features), called condition attribute set and decision attribute set, respectively, where $C \cap D = \phi$; V_a is the domain of attribute $a \in A$, and $|V_a|$ denotes the number of elements in V_a, i.e., the cardinality of V_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. Sometimes $(U,\ A = C \cup D,\ V = \bigcup_{a \in A} V_a,\ \{f_a\})_{a \in A}$ is expressed as $(U,\ C \cup D)$ for simplicity if V and f_a are understood. Without loss of generality, we suppose $D = \{d\}$ in this paper; that is, D is supposed to be composed of only one attribute. A missing value is usually denoted as "*". That is, if there exists $a \in C$ such that $* \in V_a$, then the decision system $(U,\ C \cup D)$ is an incomplete decision system.

For an incomplete decision system $IDS = (U, C \cup D)$, we define the concept of **missing value degree** (the degree of missing values) [8], denoted as ***MD*(*IDS*)**, for *IDS*:

$$MD(IDS) = \frac{\text{the number of missing attribute values}}{|U||C|}.$$

Obviously, missing value degrees have great effects on classification performance, but related studies on this problem are little reported.

Attribute-value blocks were presented by Grzymala-Busse to analyze incomplete data [8]. Here we first introduce some concepts related to attribute-value blocks.

In an incomplete system $(U,\ C \cup D)$, for any $a \in C$ and $v \in V_a$, $(\boldsymbol{a}, \boldsymbol{v})$ is said to be an **attribute-value pair**, which is an atomic formula of decision logic [9]. Let $[(a, v)]$ denote all the objects from U which can be matched with (a, v), and $[(a, v)]$ is the so-called **attribute-value (pair) block** [10]. According to the semantics of "do not care" conditions, we have:

$$[(a,v)] = \begin{cases} \{y \in U | f_a(y) = v \, \text{or} f_a(y) = *\}, & \text{if } v \neq *; \\ U, & \text{else.} \end{cases}$$

Actually, $v = f_a(x)$ for some $x \in U$, and therefore block $[(a, v)]$ is usually denoted by $K_a(x)$ or $S_a(x)$ in many studies, i.e., $K_a(x) = S_a(x) = [(a, f_a(x))]$. For $B \subseteq C$, the attribute-value block with respect to B, $K_B(x)$, is defined as follows:

$$K_B(x) = \bigcap_{a \in B} [(a,\, f_a(x))] = \bigcap_{a \in B} K_a(x).$$

Property 1. For B', $B'' \subseteq C$, if $B' \subseteq B''$, $K_{B''}(x) \subseteq K_{B''}(x)$.

2.2 Incomplete Inconsistent Decision Systems (IIDSs)

In an incomplete system $(U, C \cup D)$, since V_D does not contain missing values, D can partition U into a family of equivalence classes, which are called decision classes. In this paper, we let D_x denote the decision class that contain object x, where $x \in U$.

Definition 1. For an object $x \in U$, let $m_B(x) = \frac{|K_B(x) \cap D_x|}{|K_B(x)|}$, denoting the degree to which object x belongs to decision class D_x with respect to B, and then $\mu_B(x)$ is called the **consistency degree** of object x with respect to B.

Obviously, $0 < \mu_B(x) \leq 1$. If $\mu_C(x) = 1$, object x is said to be a **consistent object**, otherwise an **inconsistent object**. It is not difficult to find that an inconsistent object x means that block $K_C(x)$ intersects at least two different decision classes, i.e., $K_C(x) \not\subseteq D_x$.

For an incomplete decision system $(U, C \cup D)$, if U contains inconsistent objects, i.e., there exists $y \in U$ such that $\mu_C(y) < 1$, then the decision system is said to be an **incomplete inconsistent decision system** (IIDS), denoted as $IIDS = (U, C \cup D)$.

Definition 2. Let $id(IIDS)$ denote the ratio of the number of inconsistent objects to the number of all objects in U, i.e., $id(IIDS) = |\{x \in U \mid \mu_C(x) < 1\}| / |U|$, and then $id(IIDS)$ is called **inconsistency degree** of the decision system $IIDS$.

Obviously, $0 \leq id(IIDS) \leq 1$. In order to judge if an object x is consistent, we need to compute its consistency degree $\mu_B(x)$, which is possibly a time-consuming computational process because it involves set operations. The following property can make preparation for efficiently computing $\mu_B(x)$.

Property 2. For incomplete inconsistent decision system $(U, C \cup D)$ and any $x \in U$, $K_B(x) \cap D_x = \{y \in U | y \in K_B(x) \, \text{and} f_d(y) = f_d(x)\}$, and therefore $\mu_B(x) = \frac{|\{y \in U | y \in K_B(x) \wedge f_d(y) = f_d(x)\}|}{|K_B(x)|}$, where $B \subseteq C$ and $D = \{d\}$.

3 A Granulation Model Based on IIDSs

In order to compute $K_B(x)$, we generally need to traverse all objects in U, and therefore it takes the computation time of $O(|U|^2|B|)$ to compute $K_B(x)$ for all $x \in U$, which is a time-consuming computation process. However, we notice that when considering only one attribute, we can derive some useful properties to accelerate the computation process.

Definition 3. In an *IIDS* (U, $C \cup D$), for a given attribute $a \in C$, let U_a^* denote the set of all objects that $f_a(x) = *$, i.e., $U_a^* = \{x \in U | f_a(x) = *\}$; attribute a can partition $U - U_a^*$ into a family of equivalence classes, which are pairwise disjoint, and let $[x]_a$ denote an equivalence class containing x, i.e., $[x]_a = \{y \in U - U_a^* | f_a(y) = f_a(x)\}$, where $x \in U$, and let Γ_a denote such a family, i.e., $\Gamma_a = \{[x]_a | x \in U - U_a^*\}$.

Each element in Γ_a is an equivalence class. Those objects are drawn together in an equivalence class due to that they have the same attribute value on corresponding attribute. Sometimes, in order to emphasize the attribute value, we let $\Gamma_a(v)$ denote an equivalence class in Γ_a where all objects have attribute value v, i.e., $\Gamma_a(v) = \{y \in U - U_a^* | f_a(y) = v\}$.

It is not difficult to find that $\Gamma_a \cup \{U_a^*\}$ is a coverage of U; each element in $\Gamma_a \cup \{U_a^*\}$ is a subset of U, and they are also pairwise disjoint.

Property 3. Given Γ_a and U_a^*, for any $B \subseteq C$ and $x \in U$, we have

$$K_a(x) = \begin{cases} [x]_a \cup U_a^* & \mathit{if} f_a(x) \neq * \\ U & \mathit{else} \end{cases}$$

where $[x]_a \in \Gamma_a$.

Property 3 provides a method for us to use Γ_a and U_a^* to compute $K_a(x)$. We notice that $|V_a|$ almost does not increase with $|U|$, with which we can design an efficient algorithm to compute Γ_a and U_a^* for all $a \in C$. The algorithm is described as follows.

```
Algorithm 1: compute $\Gamma_a$ and $U_a^*$ for all $a \in C$
Input: ($U$, $C \cup D$), where $U = \{x_1, x_2,..., x_n\}$
Output: $\Gamma_a$ and $U_a^*$ for all $a \in C$
Begin
(1)  For each $a \in C$ do
(2)  {
(3)  Let $U_a^* = \varnothing$ and $\Gamma_a = \varnothing$;
(4)  For $i = 1$ to $n$ do  // $n = |U|$
(5)  {
(6)  If $f_a(x_i) = *$ then {let $U_a^* = U_a^* \cup \{x_i\}$; continue;}
               //Suppose $\Gamma_a = \{\Gamma_a(v_1),\ldots,\Gamma_a(v_t)\}$ at this moment, where $t = |\Gamma_a| \leq |V_a|$
(7)  Let flag = 0;
(8)  For $j = 1$ to $t$ do
(9)  If $f_a(x_i) = v_j$ then { $\Gamma_a(v_j) = \Gamma_a(v_j) \cup \{x_i\}$; let flag = 1; break;}
(10) If flag = 0 then { let $\Gamma_a(v_{t+1}) = \{x_i\}$; $\Gamma_a = \Gamma_a \cup \{\Gamma_a(v_{t+1})\}$ };
(11) }
(12) }
(13) Return $\Gamma_a$ and $U_a^*$;
End.
```

From Algorithm 1, we can find that its time complexity is $O(|C||U|t) \le O(|C||U||V_a|)$. As mentioned above, $|V_a|$ almost does not increase with $|U|$, so $|V_a|$ can be regard as a constant generally. Therefore, the time complexity of this algorithm almost approaches linear complexity $O(|C||U|)$.

In fact, Algorithm1 is to granulate each "column" for an *IIDS* and therefore to construct a granulation model for the *IIDS*. Let $\mathit{\Gamma}_a^* = \Gamma_a \cup \{U_a^*\}$, and such a granulation model is denoted as $\boldsymbol{\mathfrak{R} = (U, \{\Gamma_a^*\}_{a \in C}, D)}$ in this paper.

With the granulation model, we can compute any block $K_B(x)$ by using formula $K_B(x) = \bigcap_{a \in B} K_a(x)$. In order to quickly compute $K_B(x)$, we should know "where $K_a(x)$ is". So we construct an index structure to store the addresses of $K_a(x)$ for all $a \in C$ and $x \in U$. Such an index structure is expressed as a matrix $\boldsymbol{\psi = U \times C = [m(x, a)]_{x \in U, a \in C}}$, where $m(x, a)$ is the index or address of $\Gamma_a(v)$ and $v = f_a(x)$. The algorithm of constructing matrix ψ is described as follows.

Algorithm 2: construct matrix ψ for granulation model $\mathfrak{R}$
Input: $\mathfrak{R} = (U, \{\Gamma_a^*\}_{a \in C}, D)$
Output: $\psi = U \times C = [m(x,a)]_{x \in U, a \in C}$,

```
Begin
(1)  For each $a \in C$ do
(2)  {
(3)    For each $\theta \in \Gamma_a^*$ do
(4)    {
(5)      For each $x \in \theta$ do
(6)      {
(7)        If $f_a(x) = *$ then let $m(x,a) = $ null;// in this case, $\theta = U_a^*$
(8)        Else let $m(x,a) = loc(\Gamma_a(v))$; // the index or address of $\Gamma_a(v)$
(9)      }
(10)   }
(11) }
End.
```

Actually, Algorithm 2 is to traverse all objects in $\mathit{\Gamma}_a^*$ for all $a \in C$. We notice that $\cup \Gamma_a^* = U$ and the elements (subset) in $\mathit{\Gamma}_a^*$ are pairwise disjoint, therefore the complexity of this algorithm is exactly equal to $O(|U||C|)$, which is linear complexity.

Both granulation model $\mathfrak{R}$ and index matrix ψ are denoted as ordered pair $\boldsymbol{[\mathfrak{R}, \psi]}$. If there is no confusion, $[\mathfrak{R}, \psi]$ is also called a granulation model. The purpose of constructing $[\mathfrak{R}, \psi]$ is to provide a way to quickly compute block $K_B(x)$ for any $x \in U$, with complexity of about $O(|K_B(x)||B|)$.

4 A Granulation-Model-Based Method for Constructing Classifier

4.1 An Attribute-Value Block Based Method of Acquiring Classification Rules

A classification rule can viewed as an implication relation between different granular worlds, and each object x can derive a classification rule, which is a "bridge" between such two worlds. Firstly, let's consider the following inclusion relation: $K_B(x) \subseteq D_x$. $K_B(x)$ and D_x has their own descriptions, which are formulae of decision logic [9]. Suppose their descriptions are ρ and φ, respectively. Then, object x can derive rule $\rho \to \varphi$. According to Property 1, when removing attribute from B, $K_B(x)$ would enlarge, and therefore the generalization ability of rule $\rho \to \varphi$ would be strengthened. But its consistency degree $\mu_B(x)$ may decrease and then increase its uncertainty. Therefore the operation of removing attributes from B must be done under a certain limited condition. Now we give the concept of object reduction, which is used to acquire classification rules.

Definition 4. In an $IIDS = (U, C \cup D)$, for any object $x \in U$, $B \subseteq C$ is said to be a **reduct** of object x, if the following conditions can be satisfied: (a) $\mu_B(x) \geq \mu_C(x)$, and (b) for any $B' \subset B$, $\mu_{B'}(x) < \mu_C(x)$.

Actually, it is time-consuming to find a reduce for an object, because it needs take too much time to search each subset of B so as to satisfy condition (b). A usual method is to select some attribute from C to constitute B such that $\mu_B(x) \geq \mu_C(x)$. Such a method is known as **feature selection**, with which we can easily obtain corresponding classification rule. In the following, we give an algorithm to perform feature selection for all objects $x \in U$ and derive corresponding classification rules.

```
Algorithm 3: construct a classification rule set
Input: [$\mathfrak{R}$, $\psi$] and $(U, C \cup D)$ // suppose $U = \{x_1, x_2, \ldots, x_n\}$ and $C = \{a_1, a_2, \ldots, a_m\}$
Output: $S$   // a rule classification set
Begin
(1)  Let $S = \varnothing$;
(2)  For $i = 1$ to $n$ do //$n = |U|$
(3)  {
(4)    Let $B = C$;
(5)    For $j = 1$ to $m$ do //$m = |C|$ if $\mu_{B-\{a_j\}}(x_i) \geq \mu_C(x_i)$ then let $B = B$-$\{a_j\}$;
(6)    Use $B$ to construct $\rho \to \varphi$;
(7)    Let $S = S \cup \{\rho \to \varphi\}$;
(8)  }
(9)  Return $S$;
End.
```

In Algorithm 3, step (5) is to remove redundant attributes in C, which is actually to perform feature selection. Let $B_j = B-\{a_j\}$. According to Property 2, with [$\mathfrak{R}$, ψ],

$\mu_{B_j}(x_i)$ can be computed in the complexity of $O\left(\sum_{j=1}^{m}|K_{B_j}(x_i)||B_j|\right)$. Therefore, the complexity of Algorithm 3 is $O\left(\sum_{i=1}^{n}\sum_{j=1}^{m}|K_{B_j}(x_i)||B_j|\right)$. Suppose t is the average size of blocks $K_{B-\{a_j\}}(x_i)$ for all $a_j \in C$ and $x_i \in U$ and h is the average length of B_j, then $O\left(\sum_{i=1}^{n}\sum_{j=1}^{m}|K_{B_j}(x_i)||B_j|\right) = O(|U||C| \cdot t \cdot h)$. Generally, $t << |U|$ and $h << |C|$, so $O(|U||C| \cdot t \cdot h) << O(|U|^2|C|^2)$.

It should be pointed that Algorithm 3 can not guarantee each generated attribute subset is a reduct, but it does remove some redundant attributes from C and therefore can finish the task of feature selection.

4.2 Rule Set Minimum

Since each rule in S is induced by an object in U, these rules and objects are one to one correspondence. Usually, we let r_i denote the rule that is induced by object x_i, i.e., r_i corresponds to x_i. We notice that $|S| = |U|$ and there are many redundant rules in it. Therefore we need to further remove those redundant rules, and this process is so-called rule set minimum.

Definition 5. For rule r: $\rho \rightarrow \varphi$, let **coverage(*r*)** denote all objects which can match rule r, i.e., coverage(r) = $\{x \in U \mid x| = r\}$.

For two rules r_x: $\rho_x \rightarrow \varphi_x$ and r_y: $\rho_y \rightarrow \varphi_y$, if coverage($r_x$) $\subseteq$ coverage(r_y), then rule r_x is redundant and should be removed. Based on this consideration, we design the following algorithm to minimize the rule set S.

```
Algorithm 4: minimize a rule set
Input: [$\mathfrak{R}$, $\psi$] and $S$  // $S$ is a rule set which is generated by Algorithm 3
Output: $MS$  // $MS$ is a minimized rule set
Begin
(1)  Compute |coverage($r$)| for all $r \in S$;
(2)  Sort all rules from $S$ in a descending order by |coverage($r$)| and suppose $S$ =
     $\{r_1, r_2, \ldots, r_n\}$ after sorting;
(3)  Let $MS = \{r_1\}$;
(4)  For $i$ = 2 to $n$ do      //$n = |U|$
(5)  {
(6)    If $x_i \notin$ coverage($r_{i-1}$) then let $MS = MS \cup \{r_i\}$;   // $r_i$ is induced by $x_i$
(7)  }
(8)  Return $MS$;
End.
```

In Algorithm 4, the key operation is to compute coverage(r_{i-1}). Suppose B_{i-1} is a set of all attributes which are contained in rule r_{i-1}, and then computing coverage(r_{i-1}) is equivalent to computing block $K_{B_{i-1}}(x_{i-1})$, whose complexity is $O(|K_{B_{i-1}}(x_{i-1})||B_{i-1}|)$ by using [$\mathfrak{R}$, ψ]. Suppose the average size of blocks $K_{B_{i-1}}(x_{i-1})$ is p and the average length of $B_{i\text{-}1}$ is o, then the complexity is $O(|K_{B_1}(x_1)||B_1| + |K_{B_2}(x_2)||B_2| + \ldots + |K_{B_n}(x_n)||B_n|) = O(|U| \cdot p \cdot o)$. Generally, $p << |U|$ and $o << |C|$. Therefore $O(|U| \cdot p \cdot o) << O(|U|^2|C|)$.

4.3 A Classification Algorithm for Constructing Rule-Based Classifier

Using the above four provided algorithms, we here give a complete algorithm to acquire a rule set, which is used as a classifier to classify incomplete inconsistent data. The complete algorithm is described as follows.

```
Algorithm 5: construct a rule-based classifier
Input: $(U, C \cup D)$
Output: $MS$    // $MS$ is a rule-based classifier
Begin
(1)  Use Algorithms 1 and 2 to construct granulation model $[\mathfrak{R}, \psi]$;
(2)  Use Algorithm 3 to construct rule set $S$ by using $[\mathfrak{R}, \psi]$;
(3)  Use Algorithm 4 to minimize rule set $S$ to be $MS$ by using $[\mathfrak{R}, \psi]$ and $S$;
(4)  Return $MS$;
End.
```

As analyzed above, the complexities of Algorithms 1 and 2 are all $O(|C||U|)$, and those of Algorithms 3 and 4 are $O(|U||C| \cdot t \cdot h)$ and $O(|U| \cdot p \cdot o)$, respectively, which are much less than $O(|U|^2|C|^2)$ and $O(|U|^2|C|)$, respectively. Therefore, it can be seen that the time-consuming step is step (3), and then the complexity of Algorithm 5 is $O(|U||C| \cdot t \cdot h)$, which is much less than $O(|U|^2|C|^2)$.

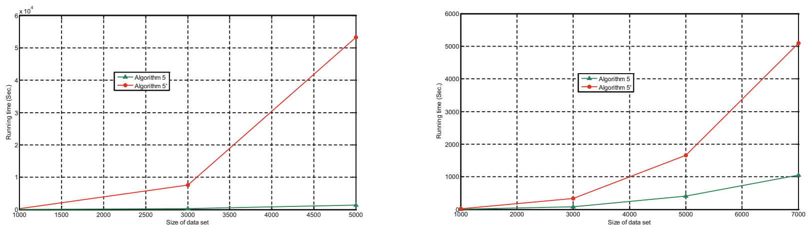
5 Experimental Analysis

In order to verify the effectiveness of the proposed methods, we conduct several experiments using UCI data sets (http://archive.ics.uci.edu/ml/datasets.html). These experiments ran on a PC equipped with a Windows 7, Intel(R) Xeon(R), CPU E5-1620v3,and 8 GB memory. The data sets are outlined in Table 1, where |U|, |C| and |Vd| stand for the numbers of samples, condition attributes, and decision classes, respectively.

For there exist missing values in incomplete decision system and the relation between objects are tolerance relation, instead of equivalence relation, we can not use sorting technique to accelerate the process of computing blocks and therefore need to compare

Table 1. Description of the four data sets.

No.	Data sets	$\|U\|$	$\|C\|$	$\|V_d\|$	$MD(IIDS)$	$id(IIDS)$
1	Voting-records	435	17	2	0.0563	0.6736
2	Tic-Tac-Toe	958	9	2	0	0
3	Mushroom	8124	22	2	0.0139	0
4	Nursery	12960	8	5	0	0



(a) Mushroom (b) Nursery

Fig. 1. Running times of Algorithm 5 and Algorithm 5′ for mushroom and nursery

x with all other objects in U when computing block $K_B(x)$, where $x \in U$. Replace the method of computing attribute-value blocks in Algorithm 5, which is based on the granulation model, with such a method of computing blocks and keep other parts uncharged. Thus we would obtain another algorithm, denoted as **Algorithm 5′**. To compare the running times for a varying number of data records, we execute Algorithm 5 and Algorithm 5′ on two data sets, Mushroom and Nursery, for four times, with randomly extracting different objects at each time, and the results are shown in Fig. 1.

Form Fig. 1 we can find that the running times of Algorithm 5′ increase much more rapidly than that of Algorithm 5. Therefore, the constructed granulation model can greatly improve computational efficiency for Algorithm 5.

Algorithm 5 consists of Algorithms 1, 2, 3 and 4. We count the time for each algorithm when they are executed on Mushroom and Nursery, and the results are shown in Table 2.

It can be seen from Table 2 that, comparing with Algorithms 3 and 4, it only takes a little time for Algorithms 1 and 2 to construct granulation model. This means that constructing granulation model cost very little but it forms the foundation for fast feature selection and building classifiers. Additionally, Table 2 also shows that Algorithm 3 is the most time-consuming algorithm.

Table 2. Running times of Algorithms 1–4 for mushroom and nursery

Data sets and the used algorithms		Size of data set			
		1000	3000	5000	7000
Mushroom	Algo. 1 + Algo. 2	0.062	0.109	0.548	0.710
	Algo. 3	20.524	293.039	1148.922	2748.864
	Algo. 4	1.602	49.749	233.802	545.356
Nursery	Algo. 1 + Algo. 2	0.042	0.058	0.164	0.358
	Algo. 3	1.955	28.818	119.987	278.489
	Algo. 4	1.382	53.047	287.059	771.875

To verify the classification performances of Algorithm 5, we utilize Voting-records, Tic-Tac-Toe, and Nursery to test Algorithm 5 using 10-fold cross-validation. The results are shown in Table 3.

Table 3. Precision and recall of Algorithm 5

Data set	Precision	Recall
Voting-records	0.9103	0.8950
Tic-Tac-Toe	0.9958	0.9969
Nursery	0.9850	0.7634

From Table 3, we can find that Algorithm 5 can have relatively high precision and recall on these data sets. This shows that the proposed algorithm has better application values. Additionally, Table 3 also shows that Algorithm 5 is suitable not only for incomplete inconsistent data but also for complete consistent data. Of course, it has better classification performances on complete consistent data than on incomplete inconsistent data.

6 Conclusion

Extracting rules from data sets and then using a rule set as a classifier to classify data is one of our purposes recent years. In this paper, oriented to incomplete inconsistent data, we first used attribute-value block technique to construct a granulation model, which actually consists of a block-based model and a index matrix; secondly, based on the constructed granulation model, an algorithm of acquiring classification rules is presented and then an algorithm of minimizing rule sets is proposed; with the proposed algorithms, we designed a classification algorithm for constructing a rule-based classifier; finally, we conducted some experiments to verify the effectiveness of the proposed algorithm. The experiment results are consistent with our theoretical analysis. Therefore, the work in this paper has a certain theoretical value and application value, and provides a new idea to classify incomplete inconsistent data.

References

1. Liu, Y., Bi, J.W., Fan, Z.P.: A method for multi-class sentiment classification based on an improved one-vs-one (OVO) strategy and the support vector machine (SVM) algorithm. Inf. Sci. **394**, 38–52 (2017)
2. Nakashima, T., Schaefer, G., Yokota, Y., et al.: A weighted fuzzy classifier and its application to image processing tasks. Fuzzy Sets Syst. **158**, 284–294 (2007)
3. Stimpfling, T., Bélanger, N., Cherkaoui, O., et al.: Extensions to decision-tree based packet classification algorithms to address new classification paradigms. Comput. Netw. **122**, 83–95 (2017)
4. Conde-Clemente, P., Trivino, G., Alonso, J.M.: Generating automatic linguistic descriptions with big data. Inf. Sci. **380**, 12–30 (2017)
5. Stefanowski, J., Tsoukiàs, A.: Incomplete information tables and rough classification. Comput. Intell. **17**, 545–566 (2001)
6. Grzymala-Busse, J.W.: A rough set approach to data with missing attribute values. In: Wang, G.-Y., Peters, J.F., Skowron, A., Yao, Y. (eds.) RSKT 2006. LNCS, vol. 4062, pp. 58–67. Springer, Heidelberg (2006). doi:10.1007/11795131_10

7. Meng, Z.Q., Shi, Z.Z.: A fast approach to attribute reduction in incomplete decision systems with tolerance relation-based rough sets. Inf. Sci. **179**, 2774–2793 (2009)
8. Meng, Z.Q., Gan, Q.L., Shi, Z.Z.: On efficient methods of computing attribute-value blocks in incomplete decision systems. Knowl.-Based Syst. **113**, 171–185 (2016)
9. Meng, Z., Gan, Q.: An attribute-value block based method of acquiring minimum rule sets: a granulation method to construct classifier. In: Shi, Z., Vadera, S., Li, G. (eds.) IIP 2016. IAICT, vol. 486, pp. 3–11. Springer, Cham (2016). doi:10.1007/978-3-319-48390-0_1
10. Grzymala-Busse, J.W., Clarka, P.G., Kuehnhausen, M.: Generalized probabilistic approximations of incomplete data. Int. J. Approx. Reas. **55**, 180–196 (2014)