Expert Systems with Applications 39 (2012) 11357–11366

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Classification based on association rules: A lattice-based approach

Loan T.T. Nguyen [a], Bay Vo [b,*], Tzung-Pei Hong [c,d], Hoang Chi Thanh [e]

[a] Faculty of Information Technology, Broadcasting College II, Ho Chi Minh, Viet Nam
[b] Information Technology College, Ho Chi Minh, Viet Nam
[c] Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung, Taiwan, ROC
[d] Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan, ROC
[e] Department of Informatics, Ha Noi University of Science, Ha Noi, Viet Nam

ARTICLE INFO

Keywords:
Classifier
Class association rules
Data mining
Lattice
Rule pruning

ABSTRACT

Classification plays an important role in decision support systems. A lot of methods for mining classification rules have been developed in recent years, such as C4.5 and ILA. These methods are, however, based on heuristics and greedy approaches to generate rule sets that are either too general or too overfitting for a given dataset. They thus often yield high error ratios. Recently, a new method for classification from data mining, called the Classification Based on Associations (CBA), has been proposed for mining class-association rules (CARs). This method has more advantages than the heuristic and greedy methods in that the former could easily remove noise, and the accuracy is thus higher. It can additionally generate a rule set that is more complete than C4.5 and ILA. One of the weaknesses of mining CARs is that it consumes more time than C4.5 and ILA because it has to check its generated rule with the set of the other rules. We thus propose an efficient pruning approach to build a classifier quickly. Firstly, we design a lattice structure and propose an algorithm for fast mining CARs using this lattice. Secondly, we develop some theorems and propose an algorithm for pruning redundant rules quickly based on these theorems. Experimental results also show that the proposed approach is more efficient than those used previously.

1. Introduction

Classification is a critical task in data analysis and decision making. For making accurate classification, a good classifier or model has to be built to predict the class of an unknown object or record. There are different types of representations for a classifier. Among them, the rule presentation is the most popular because it is similar to human reasoning. Many machine-learning approaches have been proposed to derive a set of rules automatically from a given dataset in order to build a classifier.

Recently, association rule mining has been proposed to generate rules which satisfy given support and confidence thresholds. For association rule mining, the target attribute (or class attribute) is not pre-determined. However, the target attribute must be predetermined in classification problems.

Thus, some algorithms for mining classification rules based on association rule mining have been proposed. Examples include Classification based on Predictive Association Rules (Yin and Han, 2003), Classification based on Multiple Association Rules (Li et al., 2001), Classification Based on Associations (CBA, Liu et al., 1998), Multi-class, Multi-label Associative Classification (Thabtah et al., 2004), Multi-class Classification based on Association Rules (Thabtah et al., 2005), Associative Classifier based on Maximum Entropy (Thonangi and Pudi, 2005), Noah (Guiffrida et al., 2000), and the use of the Equivalence Class Rule-tree (Vo and Le, 2008). Some researches have also reported that classifiers based on class- association rules are more accurate than those of traditional methods such as C4.5 (Quinlan, 1992) and ILA (Tolun and Abu-Soud, 1998; Tolun et al., 1999) both theoretically (Veloso et al., 2006) and with regard to experimental results (Liu et al., 1998). Veloso et al. proposed lazy associative classification (Veloso et al., 2006; Veloso et al., 2007; Veloso et al., 2011), which differed from CARs in that it used rules mined from the projected dataset of an unknown object for predicting the class instead of using the ones mined from the whole dataset. Genetic algorithms have also been applied recently for mining CARs, and some approaches have been proposed. In 2010, Chien and Chen (2010) proposed a GA-based approach to build the classifier for numeric datasets and to apply to stock trading data. Kaya (2010) proposed a Pareto-optimal for building autonomous classifiers using genetic algorithms. Qodmanan et al. (2011) proposed a GA-based method without required minimum support and minimum confidence thresholds. These algorithms were mainly based on heuristics to build classifiers.

* Corresponding author. Tel.: +84 08 39744186.
E-mail addresses: nguyenthithuyloan@vov.org.vn (L.T.T. Nguyen), vdbay@itc.edu.vn (B. Vo), tphong@nuk.edu.tw (T.-P. Hong), thanhhc@vnu.vn (H.C. Thanh).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.03.036

All the above methods focused on the design of the algorithms for mining CARs or building classifiers but did not discuss much with regard to their mining time.

Lattice-based approaches for mining association rules have recently been proposed (Vo and Le, 2009; Vo and Le, 2011a; Vo and Le, 2011b) to reduce the execution time for mining rules. Therefore, in this paper, we try to apply the lattice structure for mining CARs and pruning redundant rules quickly.

The contributions of this paper are stated as follows:

(1) A new structure called *lattice of class rules* is proposed for mining CARs efficiently; each node in the lattice contains values of attributes and their information.
(2) An algorithm for mining CARs based on the lattice is designed.
(3) Some theorems for mining CARs and pruning redundant rules quickly are developed. Based on them, we propose an algorithm for pruning CARs efficiently.

The rest of this paper is organized as follows: Some related work to mining CARs and building classifiers is introduced in Section 2. The preliminary concepts used in the paper are stated in Section 3. The lattice structure and the LOCA (Lattice of Class Associations) algorithm for generating CARs are designed in Sections 4 and 5, respectively. Section 6 proposes an algorithm for pruning redundant rules quickly according to some developed theorems. Experimental results are described and discussed in Section 7. The conclusions and future work are presented in Section 8.

2. Related work

2.1. Mining class-association rules

The Class-Association Rule (CAR) is a kind of classification rule. Its purpose is mining rules that satisfy minimum support (*minSup*) and minimum confidence (*minConf*) thresholds. Liu et al. (1998) first proposed a method for mining CARs. It generated all candidate 1-itemsets and then calculated the support for finding frequent itemsets that satisfied *minSup*. It then generated all candidate 2-itemsets from the frequent 1-itemsets in a way similar to the Apriori algorithm (Agrawal and Srikant, 1994). The same process was then executed for itemsets with more items until no candidates could be obtained. This method differed from Apriori in that it generated rules in each iteration for generating frequent k-itemsets, and from each itemset, it only generated maximum one rule if its confidence satisfied the *minConf*, where the confidence of this rule could be obtained by computing the count of the maximum class divided by the number of objects containing the left hand side. It might, however, generate a lot of candidates and scan the dataset several times, thus being quite time-consuming. The authors thus proposed a heuristic for reducing the time. They set a threshold K and only considered k-itemsets with $k \leqslant K$. In 2000, the authors also proposed an improved algorithm for solving the problem of unbalanced datasets by using multiple class minimum support values and for generating rules with complex conditions (Liu et al., 2000). They showed that the latter approach had higher accuracy than the former.

Li et al. then proposed an approach to mine CARs based on the FP-tree structure (Li et al., 2001). Its advantage was that the dataset only had to be scanned two times because the FP-tree could compress the relevant information from the dataset into a useful tree structure. It also used the tree-projection technique to find frequent itemsets quickly. Like the CBA, each itemset in the tree generated a maximum of one rule if its confidence satisfied the *minConf*. To predict the class of an unknown object, the approach found all the rules that satisfied the data and adopted the weighted χ^2 measure to determine the class.

Vo and Le (2008) then developed a tree structure called the ECR-tree (Equivalence Class Rule–tree) and proposed an algorithm named ECR-CARM for mining CARs. Their approach only scanned the dataset once and computed the supports of itemsets quickly based on the intersection of object identifications.

Some other classification association rule mining approaches have been presented in the work of Chen and Hung (2009), Coenen et al. (2007), Guiffrida et al. (2000), Hu and Li (2005), Lim and Lee (2010), Liu et al. (2008), Priss (2002), Sun et al. (2006), Thabtah et al. (2004), Thabtah et al. (2005), Thabtah et al. (2006), Thabtah (2005), Thonangi and Pudi (2005), Wang et al. (2007), Yin and Han (2003), Zhang et al. (2011) and Zhao et al. (2010).

2.2. Pruning rules and building classifiers

The CARs derived from a dataset may contain some rules that can be inferred from the others that are available. These rules need to be removed because they do not play any role in the prediction process. Liu et al. (1998) thus proposed an approach to prune rules by using the pessimistic error as C4.5 did (Quinlan, 1992). After mining CARs and pruning rules, they also proposed an algorithm to build a classifier as follows: Firstly, the mined CARs or PCARs (the set of CARs after pruning redundant rules) were sorted according to their decreasing precedence. Rule *r1* was said to have higher precedence than another rule *r2* if the confidence of *r1* was higher than that of *r2*, or their confidences were the same, but the support for *r1* was higher than that of *r2*. After that, the rules were checked according to their sorted order. When a rule was checked, all the records in a given dataset covered by the rule would be marked. If there was at least one unmarked record that could be covered by a rule *r*, then *r* was added into the knowledge base of the classifier. When an unknown object came and did not match any rule in the classifier, then a default class was assigned to it.

Another common way for pruning rules was based on the precedence and conflict concept (Chen et al., 2006; Vo and Le, 2008; Zhang et al., 2011). Chen et al. (2006) also used the concept of high precedence to point out redundant rules. Rule $r1 : Z \rightarrow c$ was redundant if there existed rule $r2 : X \rightarrow c$ such that *r2* had higher precedence than *r1*, and $X \subset Z$. Rule $r1 : Z \rightarrow c_i$ was called a conflict to rule $r2 : X \rightarrow c_j$ if *r2* had higher precedence than *r1*, and $X \subseteq Z$ ($i \neq j$). Both of redundant and conflict rules were called redundant rules in Vo and Le (2008).

3. Preliminary concepts

Let D be a set of training data with n attributes $\{A_1, A_2, \ldots, A_n\}$ and $|D|$ records (cases). Let $C = \{c_1, c_2, \ldots, c_k\}$ be a list of class labels. The specific values of attribute A and class C are denoted by the lower-case letters a and c, respectively. An itemset is first defined as follows:

Definition 1. An itemset includes a set of pairs, each of which consists of an attribute and a specific value for that attribute, denoted $<(A_{i1}, a_{i1}), (A_{i2}, a_{i2}), \ldots, (A_{im}, a_{im})>$.

Definition 2. A rule r has the form of $<(A_{i1}, a_{i1}), \ldots, (A_{im}, a_{im})> \rightarrow c_j$, where $<(A_{i1}, a_{i1}), \ldots, (A_{im}, a_{im})>$ is an itemset, and $c_j \in C$ is a class label.

Definition 3. The actual occurrence of a rule r in D, denoted $ActOcc(r)$, is the number of records in D that match r's condition.

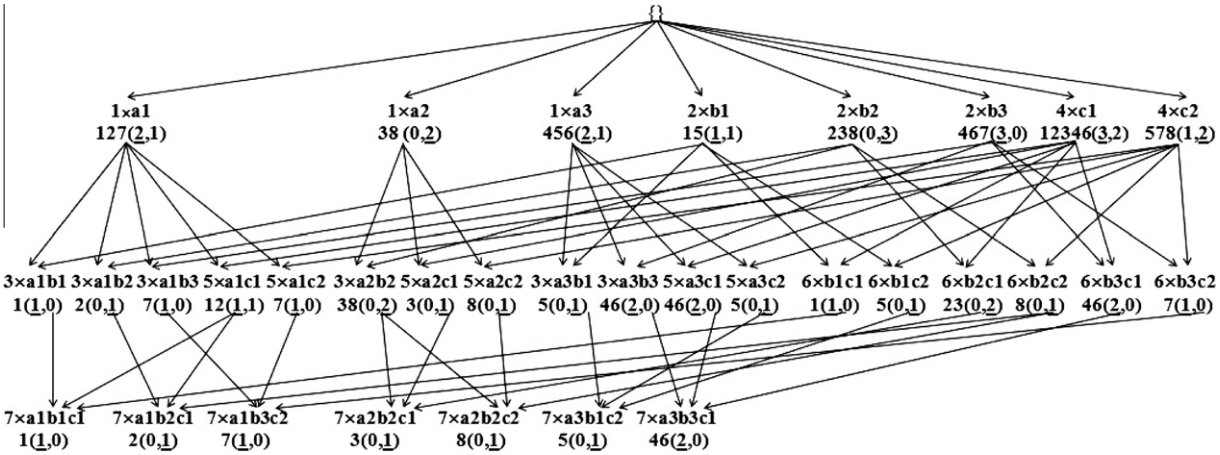

Fig. 1. A lattice structure for mining CARs.

Definition 4. The support of a rule r, denoted $Supp(r)$, is the number of records in D that match r's condition and belong to r's class.

Definition 5. The confidence of a rule r, denoted $Conf(r)$, is defined as:

$$Conf(\mathrm{r}) = \frac{Supp(r)}{ActOcc(r)}.$$

For example, assume there is a training dataset shown in Table 1 that contains eight records, three attributes, and two classes (Y and N). Both the attributes A and B have three possible values, and C has two. Consider a rule $r = \{<(A,a1)> \rightarrow Y\}$. Its actual occurrence, support and confidence are obtained as follows:

$$ActOcc(\mathrm{r}) = 3,\ \ Supp(\mathrm{r}) = 2 \text{ and } Conf(\mathrm{r}) = \frac{SuppCount(r)}{ActOcc(r)} = \frac{2}{3}.$$

Definition 6. An object identifier set of an itemset X, denoted $Obidset(X)$, is the set of object identifications in D that match X.

Take the dataset in Table 1 as an example again. The object identifier sets for the two itemsets $X1 = <(A,a2)>$ and $X2 = <(B,b2)>$ are shown as follows:

$$X1 =< (A,a2) > \text{ then } Obidset(X1)$$
$$= \{3,8\} \text{ or shortened as 38 for convenience, and}$$
$$X2 =< (B,b2) > \text{ then } Obidset(X2) = 238.$$

The object identifier set for an itemset $X3 = <(A,a2),(B,b2)>$, which is a union of $X1$ and $X2$, can be easily derived by the intersection of the above two individual object identifier sets as follows:

$$X3 =< (A,a2),(B,b2) > \text{ then } Obidset(X3)$$
$$= Obidset(X1) \cap Obidset(X2) = 38.$$

Note that $Supp(X) = |Obidset(X)|$. This is because $Obidset(\mathrm{X})$ is the set of object identifiers in D that match X.

4. The lattice structure

A lattice data structure is designed here to help mine the class-association rules efficiently. It is a lattice with vertices and arcs as explained below.

a. Vertex: Each vertex includes the following five elements:

Table 1
An example of a training dataset.

OID	A	B	C	Class
1	a1	b1	c1	Y
2	a1	b2	c1	N
3	a2	b2	c1	N
4	a3	b3	c1	Y
5	a3	b1	c2	N
6	a3	b3	c1	Y
7	a1	b3	c2	Y
8	a2	b2	c2	N

(1) values – a list of values
(2) atts – a list of attributes, each attribute contains one value in the values
(3) *Obidset* – the list of object identifiers (OIDs) containing the itemset
(4) $(c_1,c_2,\ldots,c_k)$ – where c_i is the number of records in *Obidset* which belong to class c_i, and
(5) pos – store the position of the class with the maximum count, i.e., $\text{pos} = \arg\max_{i\in[1,k]}\{c_i\}$

An example is shown in Fig. 1 constructed from the dataset in Table 1. The vertex in the first branch is $\underset{127(2,1)}{1\times a1}$, which represents that the value is $\{a1\}$ contained in objects 1, 2, 7, and two objects belong to the first class, and one belongs to the second class. The pos is 1 because the count of class Y is at its maximum (underlined at position 1 in Fig. 1).

b. Arc: An arc connects two vertices if the itemset in one vertex is the subset with one less item of the itemset in the other.

For example, in Fig. 1, the vertex containing itemset *a1* connects to the five itemsets with *a1b1*, *a1b2*, *a1b3*, *a1c1*, and *a1c2* because $\{a1\}$ is the subset with one less item. Similarly, the vertex containing *b1* connects to the vertices with *a1b1, a2b1, b1c1, b1c2*.

From the nodes (vertices) in Fig. 1, 31 CARs (with *minConf* = 60%) are derived as shown in Table 2.

Rules can be easily generated from the lattice structure. For example, consider rule 31: If $A = a3$ and $B = b3$ and $C = c1$ then class = Y (with support = 2, confidence = 2/2). It is generated from the node $\underset{46(2,0)}{7\times \text{a3b3c1}}$. The attribute is 7 (111), which means it includes three attributes with $A = a3$, $B = b3$ and $C = c1$. In addition, the values *a3b3c1* are contained in the two objects 4 and 6, and both of them belong to class = Y.

Table 2
All the CARs derived from Fig. 1 with *minConf* = 60%.

ID	Node	CARs	Supp	Conf
1	$1 \times a1$ $127(\underline{2},1)$	If $A = a1$ then class = Y	2	2/3
2	$1 \times a2$ $38(0,\underline{2})$	If $A = a2$ then class = N	2	2/2
3	$1 \times a3$ $456(\underline{2},1)$	If $A = a3$ then class = Y	2	2/3
4	$2 \times b2$ $238(0,\underline{3})$	If $B = b2$ then class = N	3	3/3
5	$2 \times b3$ $467(\underline{3},0)$	If $B = b3$ then class = Y	3	3/3
6	$4 \times c1$ $12346(\underline{3},2)$	If $C = c1$ then class = Y	3	3/5
7	$4 \times c2$ $578(1,\underline{2})$	If $C = c2$ then class = N	2	2/3
8	$3 \times a1b1$ $1(\underline{1},0)$	If $A = a1$ and $B = b1$ then class = Y	1	1/1
9	$3 \times a1b2$ $2(0,\underline{1})$	If $A = a1$ and $B = b2$ then class = N	1	1/1
10	$3 \times a1b3$ $7(\underline{1},0)$	If $A = a1$ and $B = b3$ then class = Y	1	1/1
11	$5 \times a1c2$ $7(\underline{1},0)$	If $A = a1$ and $C = c2$ then class = Y	1	1/1
12	$3 \times a2b2$ $38(0,\underline{2})$	If $A = a2$ and $B = b2$ then class = N	2	2/2
13	$5 \times a2c1$ $3(0,\underline{1})$	If $A = a2$ and $C = c1$ then class = N	1	1/1
14	$5 \times a2c2$ $8(0,\underline{1})$	If $A = a2$ and $C = c2$ then class = N	1	1/1
15	$3 \times a3b1$ $5(0,\underline{1})$	If $A = a3$ and $B = b1$ then class = N	1	1/1
16	$3 \times a3b3$ $46(\underline{2},0)$	If $A = a3$ and $B = b3$ then class = N	2	2/2
17	$5 \times a3c1$ $46(\underline{2},0)$	If $A = a3$ and $C = c1$ then class = Y	2	2/2
18	$5 \times a3c2$ $5(0,\underline{1})$	If $A = a3$ and $C = c2$ then class = N	1	1/1
19	$6 \times b1c1$ $1(\underline{1},0)$	If $B = b1$ and $C = c1$ then class = Y	1	1/1
20	$6 \times b1c2$ $5(0,\underline{1})$	If $B = b1$ and $C = c2$ then class = N	1	1/1
21	$6 \times b2c1$ $23(0,\underline{2})$	If $B = b2$ and $C = c1$ then class = N	2	2/2
22	$6 \times b2c2$ $8(0,\underline{1})$	If $B = b2$ and $C = c2$ then class = N	1	1/1
23	$6 \times b3c1$ $46(\underline{2},0)$	If $B = b3$ and $C = c1$ then class = Y	2	2/2
24	$6 \times b3c2$ $7(\underline{1},0)$	If $B = b3$ and $C = c2$ then class = Y	1	1/1
25	$7 \times a1b1c1$ $1(\underline{1},0)$	If $A = a1$ and $B = b1$ and $C = c1$ then class = Y	1	1/1
26	$7 \times a1b2c1$ $2(0,\underline{1})$	If $A = a1$ and $B = b2$ and $C = c1$ then class = N	1	1/1
27	$7 \times a1b3c2$ $7(\underline{1},0)$	If $A = a1$ and $B = b3$ and $C = c2$ then class = Y	1	1/1
28	$7 \times a2b2c1$ $3(0,\underline{1})$	If $A = a2$ and $B = b2$ and $C = c1$ then class = N	1	1/1
29	$7 \times a2b2c2$ $8(0,\underline{1})$	If $A = a2$ and $B = b2$ and $C = c2$ then class = N	1	1/1
30	$7 \times a3b1c2$ $5(0,\underline{1})$	If $A = a3$ and $B = b1$ and $C = c2$ then class = N	1	1/1
31	$7 \times a3b3c1$ $46(\underline{2},0)$	If $A = a3$ and $B = b3$ and $C = c1$ then class = Y	2	2/2

Some nodes in Fig. 1 do not generate rules because their confidences do not satisfy *minConf*. For example, the node $\frac{2 \times b1}{15(\underline{1},1)}$ has a confidence equal to 50% (< *minConf*). Note that only CARs with supports larger than or equal to the minimum support threshold are mined. From the 31 CARs in Table 2, 13 rules are obtained if *minSup* is assigned to 20%, the results for which are shown in Table 3.

The purpose of mining CARs is to generate all classification rules from a given dataset such that their supports satisfy *minSup*, and their confidences satisfy *minConf*. The details are explained in the next section.

Table 3
Rules with their supports and confidences satisfying *minSup* = 20% and *minConf* = 60%.

ID	Node	CARs	Supp	Conf
1	$1 \times a1$ $127(\underline{2},1)$	If $A = a1$ then class = Y	2	2/3
2	$1 \times a2$ $38(0,\underline{2})$	If $A = a2$ then class = N	2	2/2
3	$1 \times a3$ $456(\underline{2},1)$	If $A = a3$ then class = Y	2	2/3
4	$2 \times b2$ $238(0,\underline{3})$	If $B = b2$ then class = N	3	3/3
5	$2 \times b3$ $467(\underline{3},0)$	If $B = b3$ then class = Y	3	3/3
6	$4 \times c1$ $12346(\underline{3},2)$	If $C = c1$ then class = Y	3	3/5
7	$4 \times c2$ $578(1,\underline{2})$	If $C = c2$ then class = N	2	2/3
8	$3 \times a2b2$ $38(0,\underline{2})$	If $A = a2$ and $B = b2$ then class = N	2	2/2
9	$3 \times a3b3$ $46(\underline{2},0)$	If $A = a3$ and $B = b3$ then class = Y	2	2/2
10	$5 \times a3c1$ $46(\underline{2},0)$	If $A = a3$ and $C = c1$ then class = Y	2	2/2
11	$6 \times b2c1$ $23(0,\underline{2})$	If $B = b2$ and $C = c1$ then class = N	2	2/2
12	$6 \times b3c1$ $46(\underline{2},0)$	If $B = b3$ and $C = c1$ then class = Y	2	2/2
13	$7 \times a3b3c1$ $46(\underline{2},0)$	If $A = a3$ and $B = b3$ and $C = c1$ then class = Y	2	2/2

5. LOCA algorithm (lattice of class associations)

In this section, we introduce the proposed algorithm called LOCA for mining CARs based on a lattice. It finds the *Obidset* of an itemset by computing the intersection of the *Obidsets* of its sub-itemsets. It can thus quickly compute the supports of itemsets and only needs to scan the dataset once. The following theorem can be derived as a basis of the proposed approach:

Theorem 1. Property of vertices with the same attributes in the lattice: *Given two nodes* $\frac{att_1 \times values_1}{Obidset_1(c_{11},\ldots,c_{1k})}$ *and* $\frac{att_2 \times values_2}{Obidset_2(c_{21},\ldots,c_{2k})}$, *if* $att_1 = att_2$ *and* $values_1 \neq values_2$, *then* $Obidset_1 \cap Obidset_2 = \varnothing$.

Proof. Since $att_1 = att_2$ and $\text{values}_1 \neq \text{values}_2$, there exist a $\text{val}_1 \in \text{values}_1$ and a $\text{val}_2 \in \text{values}_2$ such that val_1 and val_2 have the same attribute but different values. Thus, if a record with OID_i contains val_1, it cannot contain val_2. Therefore, $\forall OID \in Obidset_1$, and it can be inferred that $OID \notin Obidset_2$. Thus, $Obidset_1 \cap Obidset_2 = \varnothing$.

Theorem 1 infers that, if two itemsets X and Y have the same attributes, they do not need to be combined into the itemset XY because $Supp(XY) = 0$. For example, consider the two nodes $\frac{1 \times a1}{127(1,\underline{2})}$ and $\frac{1 \times a2}{38(\underline{1},1)}$, in which $Obidset(<(A,a1)>) = 127$, and $Obidset(<(A,a2)>) = 38$. $Obidset(<(A,a1),\ (A,a2)>) = Obidset(<(A,a1)>) \cap Obidset(<(A,a2)>) = \varnothing$. Similarly, $Obidset(<(A,a1),(B,b1)>) = 1$, and $Obidset(<(A,a1),(B,b2)>) = 2$. It can be inferred that $Obidset(<(A,a1),(B,b1)>) \cap Obidset(<(A,a1),(B,b2)>) = \varnothing$ because both of these two itemsets have the same attributes AB but with different values. □

5.1. Algorithm for mining CARs

With the above theorem, the algorithm for mining CARs with the proposed lattice structure can be described as follows:

Input: *minSup*, *minConf*, and a root node L_r of the lattice which has only vertices with frequent items.
Output: CARs.
Procedure:
LOCA(L_r, *minSup*, *minConf*)
1. CARs = ∅;
2. for all $l_i \in L_r$.children do {
3. ENUMERATE_RULE (l_i, *minConf*);//generating rule that satisfies *minConf* from node l_i
4. P_i = ∅;// containing all child nodes that have their prefixes as l_i.values
5. for all $l_j \in L_r$.children, with $j > i$ do
6. if $l_i.att \neq l_j.att$ then{
7. $O.att = l_i.att \cup l_j.att$;//using the bit representation
8. $O.values = l_i.values \cup l_j.values$;
9. $O.Obidset = l_i.Obidset \cap l_j.Obidset$;
10. for all $ob \in O.Obidset$ do //computing *O.count*
11. *O.count[ob]++;*
12. $O.pos = \arg\max_{m \in [1,k]}\{O.count[m]\}$;//$k$ is the number of class
13. if $O.count[O.pos] \geqslant minSup$ then {//O is an itemset which satisfies the *minSup*
14. $P_i = P_i \cup O$;
15. Add O into the list of child nodes of l_i;
16. Add O into the list of child nodes of l_j;
17. UPDATE_LATTICE (l_i, O);//link O with its child nodes
18. }
19. }
20. LOCA (P_i, *minSup*, *minConf*); //recursively called to create the child nodes of l_i
}

The procedure **ENUMERATE_RULE**(l, *minConf*) is designed to generate the CAR from the itemset in node l with the minimim conference *minConf*. It is stated as follows:
ENUMERATE_RULE(l, *minConf*)
21. $conf = l.count[l.pos]/|l.Obidset|$;
22. if $conf \geqslant minConf$ then
23. $CARs = CARs \cup \{\text{l.itemset} \rightarrow c_{pos}(l.count[l.pos], conf)\}$;

Procedure **UPDATE_LATTICE**(l_i, O) is designed to link O with all its child nodes that have been created.
UPDATE_LATTICE(l,O)
24. for all $l_c \in l.children$ do
25. for all $l_{gc} \in l_c$.children do
26. if l_{gc}.values is a superset of O.values then
27. Add l_{gc} into the list of child nodes of O;

The above LOCA algorithm considers each node l_i with all the other nodes l_j in L_r,$j > i$ (lines 2 and 5) to generate a candidate child node O. With each pair (l_i,l_j), the algorithm checks whether l_i.att ≠ l_j.att or not (line 6). If they are different, it will compute the five elements, including *att*, *values*, *Obidset*, *count*, and *pos*, for the new node O (Lines 7–12).

Then, if the support of the rule generated by O satisfies *minSup*, i.e., $|O.count[\text{O.pos}]| \geqslant minSup$ (line 13), then node O is added to P_i as a frequent itemset (line 14). It can be observed that O is generated from l_i and l_j, so O is the child node of both l_i and l_j. Therefore, O is linked as a child node to both l_i and l_j (lines 15 and 16). Assume l_i is a node that contains a frequent k-itemset, then P_i contains all the frequent $(k+1)$-itemsets with their prefixes as l_i.values. Finally, LOCA will be recursively called with a new set P_i as its input parameter (line 20).

In addition, the procedure UPDATE_LATTICE (l_i, O) will consider each grandchild node l_{gc} of l_i with O (line 17 and lines 24 to 27), and if l_{gc}.values is a superset of O.values, then add the node l_{gc} as a child node of O in the lattice.

The procedure ENUMERATE_RULE (l, *minConf*) generates a rule from the itemset of node l. It first computes the confidence of the rule (line 21). If the confidence satisfies *minConf* (line 22), then the rule is added into CARs (line 23).

5.2. An example

Consider the dataset in Table 1 with *minSup* = 20% and *minConf* = 60%. The lattice constructed by the proposed approach is presented in Fig. 2.

The process of mining classification rules using LOCA is explained as follows: The root node (L_r = {}) contains the child nodes with single items

$$\left\{\begin{array}{cccccccc} 1\times a1 & 1\times a2 & 1\times a3 & 2\times b2 & 2\times b3 & 4\times c1 & 4\times c2 \\ 127(\underline{2},1) & 38(0,\underline{2}) & 456(\underline{2},1) & 238(0,\underline{3}) & 467(\underline{3},0) & 12346(\underline{3},2) & 578(1,\underline{2}) \end{array}\right\}$$

in the first level. It then generates the nodes of the next level. For example, consider the process of generating the node $3 \times \underset{38(0,\underline{2})}{a2b2}$. It is formed by joining node $1 \times \underset{38(0,\underline{2})}{a2}$ and node $2 \times \underset{238(0,\underline{3})}{b2}$. Firstly, the algorithm computes the intersection of {3, 8} and {2, 3, 8}, which is {3, 8} or 38 (the *Obidset* of node $3 \times \underset{38(0,\underline{2})}{a2b2}$). Because the count of the second class (count[2]) for the itemset is $2 \geqslant minSup$, a new node is created and is added into the list of the child nodes of node $1 \times \underset{38(0,\underline{2})}{a2}$ and node $2 \times \underset{238(0,\underline{3})}{b2}$. The count of this node is (0,2) because class (3) = N and class (8) = N.

Take the process of generating node $7 \times \underset{46(2,0)}{a3b3c1}$ as another example for an itemset with three items.

Node $7 \times \underset{46(\underline{2},0)}{a3b3c1}$ is generated from node $3 \times \underset{46(\underline{2},0)}{a3b3}$ and node $3 \times \underset{46(\underline{2},0)}{a3b3}$. The algorithm computes $O.Obidset = 46 \cap 46 = 46$, and adds it to the list of chidren nodes of $3 \times \underset{46(\underline{2},0)}{a3b3}$ and $5 \times \underset{46(\underline{2},0)}{a3c1}$.

From the lattice, the classification rules can be generated as follows in the recursive order:

Node $1 \times \underset{127(\underline{2},1)}{a1}$: $Conf = \frac{2}{3} \geqslant minConf \Rightarrow$ Rule 1: if $A = a1$ then class = $Y(2,\frac{2}{3})$;
Node $1 \times \underset{38(0,\underline{2})}{a2}$: $Conf = \frac{2}{2} \geqslant minConf \Rightarrow$ Rule 2: if $A = a2$ then class = $N(2,\frac{2}{2})$;
Node $3 \times \underset{38(0,\underline{2})}{a2b2}$: $Conf = \frac{2}{2} \geqslant minConf \Rightarrow$ Rule 3: if $A = a2$ and $B = b2$ then class = $N(2,\frac{2}{2})$;
Node $1 \times \underset{456(\underline{2},1)}{a3}$: $Conf = \frac{2}{3} \geqslant minConf \Rightarrow$ Rule 4: if $A = a3$ then class = $\underline{Y}(2,\frac{2}{3})$;
Node $3 \times \underset{46(\underline{2},0)}{a3b3}$: $Conf = \frac{2}{2} \geqslant minConf \Rightarrow$ Rule 5: if $A = a3$ and $B = b3$ then class = $Y(2,\frac{2}{2})$;
Node $5 \times \underset{46(\underline{2},0)}{a3c1}$: $Conf = \frac{2}{2} \geqslant minConf \Rightarrow$ Rule 6: if $A = a3$ and $C = c1$ then class = $Y(2,\frac{2}{2})$;
Node $7 \times \underset{46(\underline{2},0)}{a3b3c1}$: $Conf = \frac{2}{2} \geqslant minConf \Rightarrow$ Rule 7: if $A = a3$ and $B = b3$ and $C = c1$ then class = $Y(2,\frac{2}{2})$;
Node $2 \times \underset{238(0,\underline{3})}{b2}$: $Conf = \frac{3}{3} \geqslant minConf \Rightarrow$ Rule 8: if $B = b2$ then class = $N(3,\frac{3}{3})$;
Node $6 \times \underset{23(0,\underline{2})}{b2c1}$: $Conf = \frac{2}{2} \geqslant minConf \Rightarrow$ Rule 9: if $B = b2$ and $C = c1$ then class = $N(2,\frac{2}{2})$;

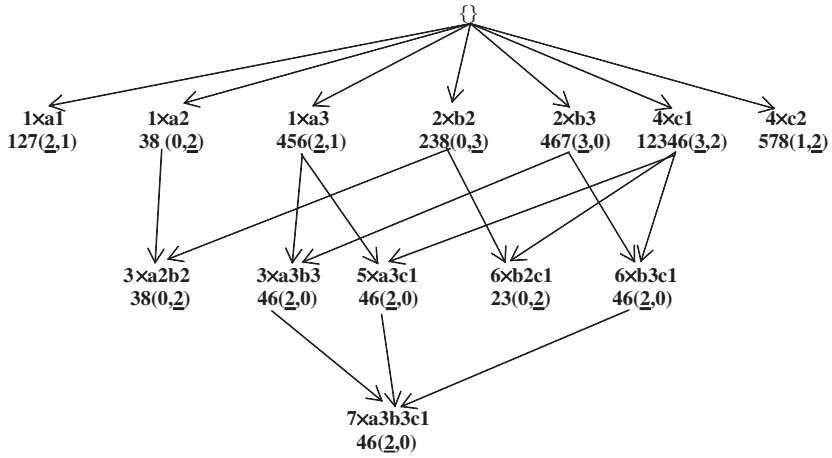

Fig. 2. The lattice constructed from Table 1 with *minSup* = 20% and *minConf* = 60%.

Table 4
Another training dataset as an example.

OID	*A*	*B*	*C*	*Class*
1	*a1*	*b1*	*c1*	*Y*
2	*a1*	*b2*	*c1*	*N*
3	*a2*	*b2*	*c1*	*N*
4	*a3*	*b3*	*c1*	*Y*
5	*a3*	*b1*	*c2*	*N*
6	*a3*	*b3*	*c1*	*Y*
7	*a1*	*b3*	*c2*	*Y*
8	*a3*	*b3*	*c2*	*N*

Node $2 \underset{467(\underline{3},0)}{\times} b3$: $Conf = \frac{3}{3} \geqslant minConf \Rightarrow$ Rule 10: if $B = b3$ then class = $Y(3, \frac{3}{3})$;

Node $6 \underset{46(\underline{2},0)}{\times} b3c1$: $Conf = \frac{2}{2} \geqslant minConf \Rightarrow$ Rule 11: if $B = b3$ and $C = c1$ then class = $Y(2, \frac{2}{2})$;

Node $4 \underset{12\,346(\underline{3},2)}{\times} c1$: $Conf = \frac{3}{5} \geqslant minConf \Rightarrow$ Rule 12: if $C = c1$ then class = $Y(3, \frac{3}{5})$;

Node $4 \underset{578(1,\underline{2})}{\times} c2$: $Conf = \frac{2}{3} \geqslant minConf \Rightarrow$ Rule 13: if $C = c2$ then class = $N(2, \frac{2}{3})$;

Thus, in total, 13 CARs are generated from the dataset in Table 1 satisfying *minSup* = 20% and *minConf* = 60%, as shown in Table 3.

6. Pruning redundant rules

LOCA generates a lot of rules, some of which are redundant because they can be inferred from the other rules. These rules may need to be removed in order to reduce storage space and to increase the prediction time. Liu et al. (1998) proposed a simple method to handle this problem. When candidate *k*-itemsets were generated in each iteration, the algorithm considered each rule with all rules that were generated preceding it to check the redundancy. Therefore, this method is time-consuming because the number of rules is very large. Thus, it is necessary to design a more efficient method to prune redundant rules. An example is given below for showing how LOCA generates redundant rules. Assume there is a dataset shown in Table 4.

With *minSup* = 20% and *minConf* = 60%, the lattice derived from the data in Table 2 is shown in Fig. 3.

It can be observed from Fig. 3 that some rules are redundant. For example, the rule *r1* (if $A = a3$ and $B = b3$ then class = Y (2, 2/3)) generated from the node $3 \underset{468(\underline{2},1)}{\times} a3b3$ is redaundant because there exists another rule *r2* (if $B = b3$ then class = Y (3, 3/4)) generated from the node $2 \underset{4678(\underline{3},1)}{\times} b3$ that is also more general than *r1*. Similarly, the rules generated from the nodes $5 \underset{58(0,\underline{2})}{\times} a3c2$, $6 \underset{23(0,\underline{2})}{\times} b2c1$ and $7 \underset{46(\underline{2},0)}{\times} a3b3c1$ are redundant. If these redundant rules are removed,

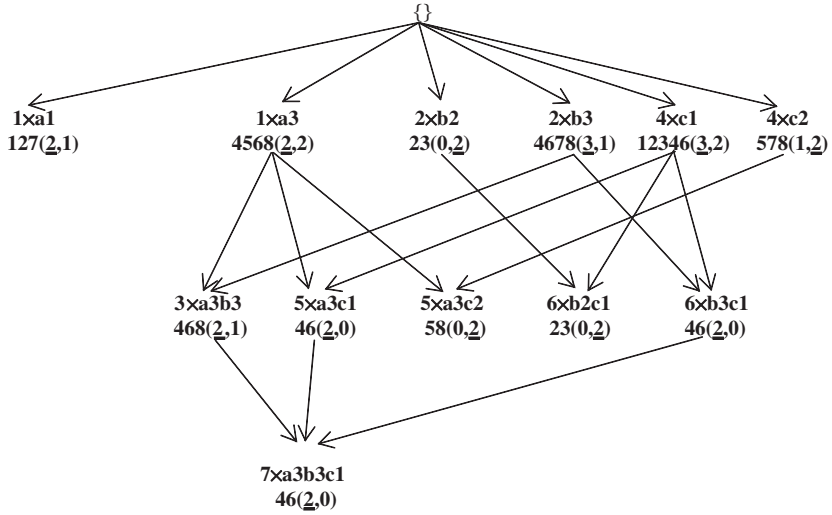

Fig. 3. The lattice constructed from Table 4 with *minSup* = 20% and *minConf* = 60%.

there remain only seven rules. Below, some definitions and theorems are given formally for pruning redundant rules.

Definition 7. – **Sub-rule** (Vo and Le, 2008) Assume there are two rules r_i and r_j, where i_i is $<(A_{i1},a_{i1}),\ldots,(A_{iu},a_{iu})> \rightarrow c_k$ and r_j is $<(B_{j1},b_{j1}),\ldots,(B_{jv},b_{jv})> \rightarrow c_l$. Rule r_i is called a sub-rule of r_j if it satisfies the following two conditions:

1. $u \leqslant v$.
2. $\forall k \in [1,u]$: $(A_{ik},a_{ik}) \in <(B_{j1},b_{j1}),\ldots,(B_{jv},b_{jv})>$.

Definition 8. – **Redundant rules** (Vo and Le, 2008) Give a rule r_i in the set of CARs from a dataset D. r_i is called a redundant rule if there is another rule r_j in the set of CARs such that r_j is a sub-rule of r_i, and $r_j \succ r_i$. From the above definitions, the following theorems can be easily derived.

Theorem 2. *If a rule r has a confidence of 100%, then all the other rules that are generated later than r and having r is a sub-rule are redundant.*

Proof. Consider r is a sub-rule of r' where r' belongs to the rule set generated later than r. To prove the theorem, we need only prove that $r \succ r'$. r has a confidence of 100%, which means that the classes of all records containing r belong to the same class. Besides, since r is a sub-rule of r', all records containing r' also contain r, which leads to all classes of records containing r' to be in the same class or the rule r' to have a confidence of 100% (1), and the support of r to be larger than or equal to the support of r' (2). From (1) and (2), we can see that $Conf(r) = Conf(r')$ and $Supp(r) \geqslant Supp(r')$, which implies that r' is a redundant rule according to Definition 8.

Based on Theorem 2, the rules with a confidence of 100% can be used to prune some redundant rules. For example, the node $\underset{467(3,0)}{2 \times b3}$ (Fig. 2) generates rule 10 with a confidence of 100%. Therefore, the other rules containing $B = b3$ may be pruned. In the above example, rules 5, 7 and 11 are pruned. Because all the rules generated from the child nodes of a node l that contains a rule with a confident of 100% are redundant, node l can thus be deleted after storing the generated rule. Some search space and memory to store nodes can thus be reduced. □

Theorem 3. *Given two rules r_i and r_j, generated from the node* $\underset{Obidset_1(c_{11},\ldots,c_{1k})}{att_1 \times values_1}$ *and the node* $\underset{Obidset_2(c_{21},\ldots,c_{2k})}{att_2 \times values_2}$*, respectively, if $values_1 \subset values_2$ and Conf (r1) $\geqslant$ Conf (r2), then rule r2 is redundant.*

Proof. Since $values_1 \subset values_2$, *r1* is a sub-rule of *r2* (according to Definition 7). Additionally, since $Conf(r1) \geqslant Conf(r2) \Rightarrow r1 \succ r2$. *r2* is thus redundant (according to Definition 8). □

6.1. Algorithm for pruning rules

In this section, we present an algorithm which is an extension of LOCA, to prune redundant rules. According to Theorem 2, if a node contains a rule with a confidence of 100%, it must be deleted and does not need to be further explored from the node. Additionally, if a rule is generated with a confidence <100%, it must be checked to determine whether it is redundant or not using Theorem 3. The PLOCA procedure is stated as follows:

Input: *minSup*, *minConf*, a root node L_r of lattice which has only vertices with frequent items.
Output: A set of class-association rules (called pCARs) with redundant rules pruned.
Procedure:
PLOCA(L_r, *minSup*, *minConf*)
1. *pCARs* = ∅;
2. for all $l_i \in L_r$.children do
3. **ENUMERATE_RULE_1**(l_i);//generating rule with 100% confidence and deleting some nodes
4. for all $l_i \in L_r$.children do {
5. P_i = ∅;
6. for all $l_j \in L_r$.children, with $j > i$ do
7. if l_i.att ≠ q l_j.att then {
8. O.att = l_i.att ∪ l_j.att;
9. O.values = l_i.values ∪ l_j.values;
10. $O.Obidset = l_i.Obidset \cap l_j.Obidset$;
11. for all $ob \in O.Obidset$ do
12. $O.count[ob]$ + +;
13. $O.pos = \arg\max_{m \in [1,k]} \{O.count[m]\}$;
14. if O.count[O.pos] $\geqslant$ *minSup* then
15. if $\frac{O.count[O.pos]}{|O.Obidset|} <$ *minConf* or $\frac{O.count[O.pos]}{|O.Obidset|} \leqslant \frac{l_i.count[l_i.pos]}{|l_i.Obidset|}$ or $\frac{O.count[O.pos]}{|O.Obidset|} \leqslant \frac{l_j.count[l_j.pos]}{|l_j.Obidset|}$ then
16. O.hasRule = false;//O will not generate rule
17. else O.hasRule = true;//O will be used to generate rule
18. $P_i = P_i \cup O$;
19. Add O to the list of child nodes of l_i;
20. Add O to the list of child nodes of l_j;
21. **UPDATE_LATTICE**(l_i, O);
22. **PLOCA**(P_i, *minSup*, *minConf*);
23. if l_i.hasRule = true then
24. pCARs = pCARs ∪ {l_i.itemset → $c_{li.pos}$(l_i.count[l.pos],l_i.count[l_i.pos]/|$l_i.Obidset$|)};
25.
ENUMERATE_RULE_1(l)
25. *conf* = l.*count*[l.*pos*]/|l.*Obidset*|;
26. if conf = 1.0 then
27. *pCARs* = *pCARs* ∪{l.*itemset* → $c_{l.pos}$(l.*count*[l.pos],*conf*)};
28. *Delete* node l;

The PLOCA algorithm is based on theorems 2 and 3 to prune redundant rules quickly. It differs from LOCA in the following ways:

(i) In the case of a confidence = 100%, the procedure ENUMERATE_RULE_1 can delete the node which generates the rule (line 27). Thus, this procedure will not generate any candidate superset which has the itemset of this rule as its prefix.
(ii) For rules with a confidence < 100%, Theorem 3 is used to remove redundant rules (line 15). When two nodes l_i and l_j are joined to form a new node O, if $\frac{O.count[O.pos]}{|O.Obidset|} \leqslant \frac{l_i.count[l_i.pos]}{|l_i.Obidset|}$ or $\frac{O.count[O.pos]}{|O.Obidset|} \leqslant \frac{l_j.count[l_j.pos]}{|l_j.Obidset|}$, the rules generated by the node O are redundant. In this case, the algorithm will assign *O.hasRule* as false (Line 16), meaning no rule needs to be generated from the node O. Otherwise, *O.hasRule* is set true (Line 17), meaning a rule needs to be generated from the node O.
(iii) Procedure UPDATE_LATTICE (l_i, O) will also consider each child node l_{gc} of l_i.children, if O.itemset ⊂ l_{gc}.itemset, then l_{gc} is the child node of $O \Rightarrow$ Add l_{gc} into the list of child nodes of O. We additionally consider whether the rule generated by l_{gc} is redundant or not by using Theorem 3.

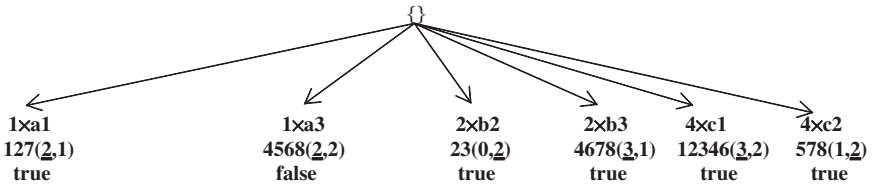

Fig. 4. The first level of the LECR structure in this example.

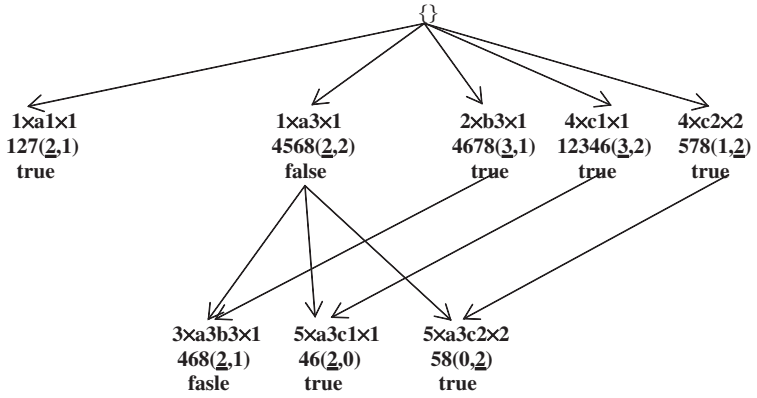

Fig. 5. Nodes generated from the node $1 \times \underset{4568(2,2)}{a3} \times 1$.

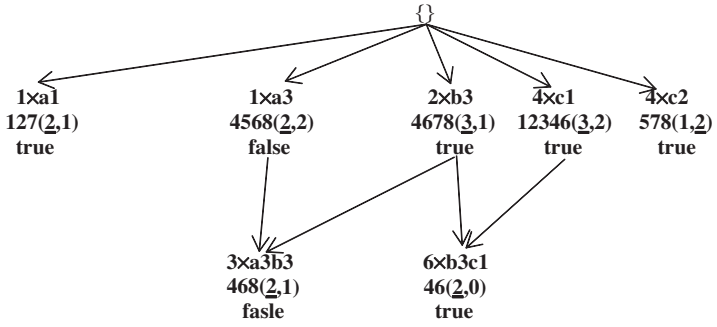

Fig. 6. Final lattice with *minSup* = 20% and *minConf* = 60%.

(iv) Procedure ENUMERATE_RULE_1 generates rules with a confidence of 100% only. The algorithm still has to generate all the other rules from the lattice. This can be easily done by checking the variable *hasRule* in node l_i. If *hasRule* is true, then a rule needs to be generated (lines 22–23).

6.2. An example

Consider the dataset given in Table 4 with *minSup* = 20% and *minConf* = 60%. The process for constructing the lattice by the PLOCA algorithm is done, and the results for the growth of the first level are shown in Fig. 4.

It can be observed from Fig. 4 that node $2 \times \underset{23(0,2)}{b2}$ generates the rule *r1* (if *B* = *b2* then class = *N*) with a confidence of 100%. The node is thus deleted, and no more exploration from the node is needed. Besides, the variable *hasRule* of the node $1 \times \underset{127(2,1)}{a1}$ is true because $count[\text{pos}]/|Obidset| = 2/3 \geqslant minConf$. The variable *hasRule* of the node $1 \times \underset{4568(2,2)}{a3}$ is false because $count[\text{pos}]/|Obidset| = 2/4 < minConf$. After the nodes for generating rules with a confidence of 100% on level 1 are removed, the the lattice up to level 2 is shown in Fig. 5.

Consider node $l_1 = 1 \times \underset{127(2,1)}{a1}$: l_1 will join with all nodes following it to create the set P_1. Because $|Obidset\ (l_1) \cap Obidset(l_j)| < 2,\ \forall j > 1$, $P_1 = \varnothing$.

Consider node $l_2 = 1 \times \underset{4568(2,2)}{a3}$, l_2 will join with all nodes following it to create the set P_2:

- With node $2 \times \underset{4678(3,1)}{b3}$ to create new node $3 \times \underset{468(2,1)}{a3b3} \Rightarrow P_2 = \left\{3 \times \underset{468(2,1)}{a3b3}\right\}$.

Table 5
The characteristics of the experimental datasets.

Dataset	*#Attrs*	*#Classes*	*#Distinct values*	*#Objs*
Breast	12	2	737	699
German	21	2	1077	1000
Lymph	18	4	63	148
Poker-hand	11	10	95	1 000 000
Led7	8	10	24	3200
Vehicle	19	4	1434	846

Table 6
Experimental results for different minimum supports.

Dataset	*minSup* (%)	#PCARs	Time (s)		$\frac{(2)}{(1)} \times 100\%$
			pCARM (1)	PLOCA (2)	
Breast	1	827	0.1	0.1	100
	0.5	1430	0.12	0.11	91.67
	0.3	2878	0.16	0.13	81.25
	0.1	6180	0.19	0.17	89.47
German	4	36431	2.31	1.43	61.9
	3	64512	2.48	2.03	81.85
	2	135266	5.22	3.25	62.26
	1	406384	13.16	6.13	46.58
Lymph	4	120617	3.97	1.89	47.61
	3	175160	4.03	3.64	90.32
	2	422734	5.84	4.15	71.06
	1	743499	13.43	5.56	41.4
Poker-hand	5	20	46.6	14	30.04
	4	68	46.75	42.94	91.85
	3	108	48.94	45.19	92.34
	2	108	142.48	50.73	35.6
Led7	1	503	0.27	0.25	92.59
	0.5	880	0.28	0.26	92.86
	0.3	901	0.29	0.27	93.1
	0.1	1023	0.31	0.28	90.32
Vehicle	1	492	0.56	0.48	85.71
	0.5	3083	1.03	0.74	71.84
	0.3	7100	1.37	0.98	71.53
	0.1	214231	2.71	1.95	71.96

- With node $\underset{12346(\underline{3},2)}{4 \times c1}$ to create new node $\underset{46(\underline{2},0)}{5 \times a3c1} \Rightarrow P_2 = \left\{ \underset{468(\underline{2},1)}{3 \times a3b3}, \underset{46(\underline{2},0)}{5 \times a3c1} \right\}$.
- With node $\underset{578(1,\underline{2})}{4 \times c2}$ to create new node $\underset{58(0,\underline{2})}{5 \times a3c2} \Rightarrow P_2 = \left\{ \underset{468(\underline{2},1)}{3 \times a3b3}, \underset{46(\underline{2},0)}{5 \times a3c1}, \underset{58(0,\underline{2})}{5 \times a3c2} \right\}$.

Consider each node in P_2 (Fig. 5) in which the variable *hasRule* of the node $\underset{468(\underline{2},1)}{3 \times a3b3}$ is false because the confidence of the rule generated by it is 2/3, which is smaller than the confidence of the rule generated by the node $\underset{4678(\underline{3},1)}{2 \times b3}$(3/4). Another node $\underset{46(\underline{2},0)}{5 \times a3c1}$ will generate rule *r2* (if *A* = *a3* and *C* = *c1* then class = *Y* (2, 0)), and it is removed since it has 100% confidence. Similarly, the node $\underset{58(0,\underline{2})}{5 \times a3c2}$ generates the rule *r3* (if *A* = *a3* and *C* = *c2* then class = *N*(0, 2)), and it is removed.

The final lattice after the execution is shown in Fig. 6.

Next, the algorithm will traverse the lattice to generate all the rules with *hasRule* = true. Thus, after pruning redundant rules, we have the following results:

Rule r1: if *B* = *b2* then class = *N* (2, 1);
Rule r2: if *A* = *a3* and *C* = *c1* then class = *Y*(2, 1);
Rule r3: if *A* = *a3* and *C* = *c2* then class = *N*(2, 1);
Rule r4: if *A* = *a1* then class = *Y*(2, 2/3);
Rule r5: if *B* = *b3* then class = *Y*(3, 3/4);
Rule r6: if *C* = *c1* then class = *Y*(3, 3/5);
Rule r7: if *C* = *c2* then class = *N*(2, 2/3);
Rule r8: if *B* = *b3* and *C* = *c1* then class = *Y*(2, 1).

7. Experimental results

The algorithms used in the experiments were coded on a personal computer with C#2008, Windows 7, Centrino 2 × 2.53 GHz, and 4MBs RAM. The experimental results were tested in the datasets obtained from the UCI Machine Learning Repository (http://mlearn.ics.uci.edu). Table 5 shows the characteristics of the experimental datasets.

The experimental datasets have different features. The Breast, German and Vehicle datasets have many attributes and distinctive items but have few numbers of objects (or records). The Led7 dataset has a few attributes, distinctive items and number of objects. The Poker-hand dataset has a few attributes and distinctive items, but has a large number of objects.

Experiments were made to compare the number of PCARs and the execution time along with different minimum supports for the same *minConf* = 50%. The results are shown in Table 6. It can be found from the table that the datasets with more numbers of attributes generated more rules and needed longer time.

The experimental results in Table 6 show that PLOCA was more efficient than pCARM with regard to mining time. For example, with the Lymph dataset (*minSup* = 1%, *minConf* = 50%), the number of rules generated was 743,499. The mining time using pCARM was 13.43 and using PLOCA was 5.56, and the ratio was found to be 41.4%.

8. Conclusions and future work

In this paper, we proposed a lattice-based approach for mining class-association rules, and two algorithms for efficient mining CARs and PCARs were presented, respectively. The purpose of using the lattice structure was to check easily whether a rule generated from a lattice node was redundant or not by comparing it with all its parent nodes. If there was a parent node such that the confidence of the rule generated by the parent node was found to be higher than that generated by the current node, then the rule generated by the current node was determined to be redundant. Based on this approach, a generated rule is not necessarily checked with a lot of other rules that have been generated. Therefore, the mining time can be greatly reduced. It is additionally not necessary to check whether two elements have the same prefix when using the lattice. Therefore, using PLOCA is often faster than using pCARM.

There have been a lot of interestingness measures proposed for evaluating association rules (Vo and Le, 2011b). In the future, we will study how to apply these measures in CARs/PCARs and discuss the impact of these interestingness measures with regard to the accuracy of the classifiers built. Because mining association rules from incremental datasets has been developed in recent years (Gharib et al., 2010; Hong and Wang, 2010; Hong et al., 2009; Hong et al., 2011; Lin et al., 2009), we will also attempt to apply incremental mining to maintain CARs for dynamic datasets.

References

Agrawal, R., & Srikant, R. (1994). Fast algorithm for mining association rules. In *The international conference on very large databases* (pp. 487–499). Santiago the Chile, Chile.

Chen, Y. L., & Hung, L. T. H. (2009). Using decision trees to summarize associative classification rules. *Expert Systems with Applications, 36*(2), 2338–2351.

Chen, G., Liu, H., Yu, L., Wei, Q., & Zhang, X. (2006). A new approach to classification based on association rule mining. *Decision Support Systems, 42*(2), 674–689.

Chien, Y. W. C., & Chen, Y. L. (2010). Mining associative classification rules with stock trading data – A GA-based method. *Knowledge-Based Systems, 23*(6), 605–614.

Coenen, F., Leng, P., & Zhang, L. (2007). The effect of threshold values on association rule based classification accuracy. *Data & Knowledge Engineering, 60*(2), 345–360.

Gharib, T. F., Nassar, H., Taha, M., & Abraham, A. (2010). An efficient algorithm for incremental mining of temporal association rules. *Data & Knowledge Engineering, 69*(8), 800–815.

Guiffrida, G., Chu, W. W., & Hanssens, D. M. (2000). Mining classification rules from datasets with large number of many-valued attributes. In *The 7th International Conference on Extending Database Technology: Advances in Database Technology (EDBT'06)* (pp. 335–349), Munich, Germany.

Hong, T. P., Lin, C. W., & Wu, Y. L. (2009). Maintenance of fast updated frequent pattern trees for record deletion. *Computational Statistics and Data Analysis, 53*(7), 2485–2499.

Hong, T. P., & Wang, C. J. (2010). An efficient and effective association-rule maintenance algorithm for record modification. *Expert Systems with Applications, 37*(1), 618–626.

Hong, T. P., Wang, C. Y., & Tseng, S. S. (2011). An incremental mining algorithm for maintaining sequential patterns using pre-large sequences. *Expert Systems with Applications, 38*(6), 7051–7058.

Hu, H., & Li, J. (2005). Using association rules to make rule-based classifiers robust. In *The 16th Australasian Database Conference* (pp. 47–54). Newcastle, Australia.

Kaya, M. (2010). Autonomous classifiers with understandable rule using multi-objective genetic algorithms. *Expert Systems with Applications, 37*(4), 3489–3494.

Li, W., Han, J., & Pei, J. (2001). CMAR: Accurate and efficient classification based on multiple class-association rules. In *The 1st IEEE international conference on data mining* (pp. 369–376). San Jose, California, USA.

Lim, A. H. L., & Lee, C. S. (2010). Processing online analytics with classification and association rule mining. *Knowledge-Based Systems, 23*(3), 248–255.

Lin, C. W., Hong, T. P., & Lu, W. H. (2009). The Pre-FUFP algorithm for incremental mining. *Expert Systems with Applications, 36*(5), 9498–9505.

Liu, B., Hsu, W., & Ma, Y. (1998). Integrating classification and association rule mining. In *The 4th international conference on knowledge discovery and data mining* (pp. 80–86). New York, USA.

Liu, B., Ma, Y., & Wong, C. K. (2000). Improving an association rule based classifier. In *The 4th European conference on principles of data mining and knowledge discovery* (pp. 80–86). Lyon, France.

Liu, Y. Z., Jiang, Y. C., Liu, X., & Yang, S. L. (2008). CSMC: A combination strategy for multiclass classification based on multiple association rules. *Knowledge-Based Systems, 21*(8), 786–793.

Priss, U. (2002). A classification of associative and formal concepts. In *The Chicago Linguistic Society's 38th Annual Meeting* (pp. 273–284). Chicago, USA.

Qodmanan, H. R., Nasiri, M., & Minaei-Bidgoli, B. (2011). Multi objective association rule mining with genetic algorithm without specifying minimum support and minimum confidence. *Expert Systems with Applications, 38*(1), 288–298.

Quinlan, J. R. (1992). *C4.5: program for machine learning*. Morgan Kaufman.

Sun, Y., Wang, Y., & Wong, A. K. C. (2006). Boosting an associative classifier. *IEEE Transactions on Knowledge and Data Engineering, 18*(7), 988–992.

Thabtah, F. (2005). Rule pruning in associative classification mining. In *The 11th international business information management (IBIMA 2005)*. Lisbon, Portugal.

Thabtah, F., Cowling, P., & Hammoud, S. (2006). Improving rule sorting, predictive accuracy and training time in associative classification. *Expert Systems with Applications, 31*(2), 414–426.

Thabtah, F., Cowling, P., & Peng, Y. (2004). MMAC: A new multi-class, multi-label associative classification approach. In *The 4th IEEE international conference on data mining* (pp. 217–224). Brighton, UK.

Thabtah, F., Cowling, P., & Peng, Y. (2005). MCAR: Multi-class classification based on association rule. In *The 3rd ACS/IEEE international conference on computer systems and applications* (pp. 33–39). Tunis, Tunisia.

Thonangi, R., & Pudi, V. (2005). ACME: An associative classifier based on maximum entropy principle. In *The 16th International Conference Algorithmic Learning Theory, LNAI 3734* (pp. 122–134). Singapore.

Tolun, M. R., & Abu-Soud, S. M. (1998). ILA: An inductive learning algorithm for production rule discovery. *Expert Systems with Applications, 14*(3), 361–370.

Tolun, M. R., Sever, H., Uludag, M., & Abu-Soud, S. M. (1999). ILA-2: An inductive learning algorithm for knowledge discovery. *Cybernetics and Systems, 30*(7), 609–628.

Veloso, A., Meira Jr., W., & Zaki, M. J. (2006). Lazy associative classification. In *The 2006 IEEE international conference on data mining (ICDM'06)* (pp. 645–654). Hong Kong, China.

Veloso, A., Meira Jr., W., Goncalves, M., & Zaki, M. J. (2007). Multi-label lazy associative classification. In *The 11th European conference on principles of data mining and knowledge discovery* (pp. 605–612). Warsaw, Poland.

Veloso, A., Meira, W., Jr., Goncalves, M., Almeida, H. M., & Zaki, M. J. (2011). Calibrated lazy associative classification. *Information Sciences, 181*(13), 2656–2670.

Vo, B., & Le, B. (2008). A novel classification algorithm based on association rule mining. In *The 2008 Pacific Rim Knowledge Acquisition Workshop (Held with PRICAI'08), LNAI 5465* (pp. 61–75). Ha Noi, Viet Nam.

Vo, B., & Le, B. (2009). Mining traditional association rules using frequent itemsets lattice. In *The 39th international conference on computers & industrial engineering* (pp. 1401–1406). July 6–8, Troyes, France.

Vo, B., & Le, B. (2011a). Mining minimal non-redundant association rules using frequent itemsets lattice. *International Journal of Intelligent Systems Technology and Applications, 10*(1), 92–106.

Vo, B., & Le, B. (2011b). Interestingness measures for association rules: Combination between lattice and hash tables. *Expert Systems with Applications, 38*(9), 1630–11640.

Wang, Y. J., Xin, Q., & Coenen, F. (2007). A novel rule ordering approach in classification association rule mining. In *International conference on machine learning and data mining, LNAI 4571* (pp. 339–348). Leipzig, Germany.

Yin, X., & Han, J. (2003). CPAR: Classification based on predictive association rules. In *SIAM International Conference on Data Mining (SDM'03)* (pp. 331–335). San Francisco, CA, USA.

Zhang, X., Chen, G., & Wei, Q. (2011). Building a highly-compact and accurate associative classifier. *Applied Intelligence, 34*(1), 74–86.

Zhao, S., Tsang, E. C. C., Chen, D., & Wang, X. Z. (2010). Building a rule-based classifier – A fuzzy-rough set approach. *IEEE Transactions on Knowledge and Data Engineering, 22*(5), 624–638.