A Generalized Apriori Algorithm

Baoqing Jiang[1,2], Fengbin Zheng[1] and Yang Xu[2]

[1]School of Computer and Information Engineering, Henan University, Kaifeng, Henan 475001, China
[2]Center of Intelligent Control and Development, Southwest Jiaotong University, Chengdu, Sichuan 610031, China

Abstract

The famous Apriori algorithm for finding the set of all frequent itemsets was presented by R. Agrawal and R. Srikant in their 1994 paper. The set of all frequent itemsets is a lower segment of the power set lattice of the set of all items. In this paper, we generalize the power set lattice of the set of all items to a direct product of finite finite-chains and discuss a generalized Apriori algorithm GenApriori for discovering a lower segment of the direct product of finite finite-chains. Some propositions, examples and experiments illustrate the correctness of the algorithm GenApriori.

Keywords: Data mining; Apriori algorithm; direct product; lower segment; maximal element

1. Introduction

The problem of mining association rules from large databases has been subject of numerous studies. Some of them focus on developing faster algorithms for the classical Apriori algorithm [1]. Another direction is to define rules that modify some conditions of the classical rules to adapt to new applications. Flip Korn focused on real-valued data such as dollar amounts spent by customers on products in a transaction database and proposed *Ratio Rules*. Ratio rules can achieve more compact descriptions and so can better perform extrapolations and predictions, if the data points are linearly correlated. But the data in transaction database are not always like that. So we discuss a weaker form called *Weak Ratio Rules*. In weak ratio rules problem, the transaction database can be thought as a nonnegative real-valued matrix, in which every row corresponds to a transaction and every column corresponds to an item. Given minimum support threshold *ms* and minimum confidence threshold *mc*, an important subset *S* of weak ratio rules is just a lower segment of a direct product of finite finite-chains. In order to get all the elements of *S*, we investigated the famous Apriori algorithm and found that it is a generalized Apriori algorithm problem.

Let $I=\{i_1, i_2, \ldots, i_m\}$ be the set of all items. Let $P(I)$ be the set of all subsets on I. Then $P(I)$ is a Boolean lattice *w.r.t.* inclusion relation $\subseteq$. Let L be the set of all frequent itemsets. Then L is a lower segment [2] of $P(I)$. Here we suppose that the empty set $\varnothing$ is frequent. As we know, $P(I) \cong \{0, 1\}^m$, where $\{0, 1\}$ is the chain containing only the least element 0 and the greatest element 1, $m = |I|$ and $\{0, 1\}^m$ is the direct product. So, we can say that the Apriori algorithm is a process for finding all elements, layer by layer, of a lower segment of $\{0, 1\}^m$.

In this paper, we will generalize $\{0, 1\}^m$ to $V_1 \times V_2 \times \ldots \times V_m$, where, for any $i = 1, 2, \ldots, m$, $V_i = \{0, 1, \ldots, n_i\}$ is the chain with the usual linear order $\leq$ (less than or equal to) of natural number. And in section 5, we will propose an algorithm for finding all elements of a lower segment of $V_1 \times V_2 \times \ldots \times V_m$.

2. Preliminaries

We introduce several basic concepts which will be used throughout the paper.

Definition 2.1 Let P be a poset and $x, y \in P$. If $x < y$ and there is no element z in P such that $x < z < y$ then we say that y is an **upper adjacency** of x, or x is a **lower adjacency** of y.

Definition 2.2 Let $V_i=\{0,1,\ldots,n_i\}$ be the chain with the usual linear order $\leq$ (less than or equal to) of natural number, $i=1, 2,\ldots,m$. Then the direct product $V_1 \times V_2 \times \ldots \times V_m$, *i.e.*, $\{0,1,\ldots,n_1\} \times \{0,1,\ldots,n_2\} \times \ldots \times \{0,1,\ldots,n_m\}$, is called a **position lattice**, denoted by $\mathrm{Pos}(n_1,n_2,\ldots,n_m)$, and n_i is called i **component upper bound** of the position lattice for any $i=1, 2,\ldots,m$.

Definition 2.3 [2] Let P be a poset, and S be a subset of P. If every element s in S such that

"$p \leq s$ implies $p \in S$", for any $p \in P$,

then S is called a **lower segment** of P. We let empty set $\varnothing$ be a lower segment of P.

By the Definition 2.3, we know that a lower segment consists of all the elements which have a property that if one element has the property then all the elements which are less than or equal to the element

also have the property. For instance, all the frequent itemsets constitute a lower segment of $P(I)$ and all the hind parts of association rules generated by a given frequent itemset F constitute a lower segment of $P(F)$.

Definition 2.4 [3] Let B be a subset of a poset P and $b \in B$. If there is no element c in B such that $b < c$, then b is called a **maximal element** of B.

Obviously, a finite lower segment can be determined uniquely by its maximal elements.

3. Layers of a lower segment

An element in Pos(n_1, n_2, …, n_m) can be written as $\langle s_1, s_2, \dots, s_m\rangle$, where $0 \le s_i \le n_i$ for any $i = 1, 2, \dots, m$. For convenience, we call the element α as vector and use $\alpha(i)$ to mean s_i, the component i of vector α.

Definition 3.1 For any $k = 0, 1, \dots, \sum_{i=1}^{m} n_i$, the subset $G_k \overset{\Delta}{=} \{\langle s_1, s_2, \dots, s_m\rangle \mid 0 \le s_i \le n_i, i = 1,2,\dots,m, \sum_{i=1}^{m} s_i = k\}$ is called the ***k*-grade** or ***k*-layer** of Pos(n_1,n_2,…,n_m).

Obviously, any two different elements in G_k are incomparable, G_k is the set of all upper adjacencies of all elements in G_{k-1}, and G_{k-1} is the set of all lower adjacencies of all elements in G_k. For example, Pos(4,3,4) has 12 layers:

$G_{11} = \{434\}$; ($\langle x, y, z\rangle$ denoted simply by xyz)

…

$G_1 = \{001, 010, 100\}$;

$G_0 = \{000\}$.

Let D be a set of transactions and $I = \{i_1, i_2, \dots, i_m\}$ be all items of D. A subset X, of I, can be expressed as a m-dimensional vector $\langle s_1, s_2, \dots, s_m\rangle$, for any $j = 1, 2, \dots, m$, $s_j = 1$ if and only if $i_j \in X$; $s_j = 0$ if and only if $i_j \notin X$. At this point, $P(I)$ is isomorphic with position lattice $\{0, 1\}^m$. In the famous Apriori algorithm, a k-itemset is an element of k-layer of $\{0, 1\}^m$. L_k, the set of all frequent k-itemsets, is a subset of k-layer of $\{0, 1\}^m$. The method for finding L_k is that: generating C_k, the set of all candidate k-itemsets, from L_{k-1} firstly, then traversing D to judge which elements in C_k belong to L_k.

In the following, we generalize the process of generating C_k by L_{k-1} to Pos(n_1,n_2,…,n_m).

Definition 3.2 Suppose $R_{k-1} \subseteq G_{k-1}$, $\alpha_k \in G_k$, $1 \le k \le \sum_{i=1}^{m} n_i$. If all lower adjacencies of α_k belong to R_{k-1}, then α_k is called a **closed upper adjacency** of R_{k-1}. Let $P_c(R_{k-1})$ denote the set of all closed upper adjacencies of R_{k-1}.

Obviously, for any $k (1 \le k \le \sum_{i=1}^{m} n_i)$, $P_c(G_{k-1}) = G_k$.

Definition 3.3 Let $\alpha_{k-1}, r_1, r_2 \in G_{k-1}$, $R_{k-1} \subseteq G_{k-1}$, $2 \le k \le \sum_{i=1}^{m} n_i$.

If there exists a $j \in \{1, 2, \dots, m\}$ such that $\alpha_{k-1}(j) < n_j \wedge k$ and for any $i \ne j, \alpha_{k-1}(i) = 0$, then we say that α_{k-1} can generate 1-type upper adjacency which is a vector whose component j is $\alpha_{k-1}(j)+1$ and other component are 0, and we use $p_1(\alpha_{k-1})$ to denote the **1-type upper adjacency**.

If there exist $i_1, i_2 \in \{1, 2, \dots, m\}$, $i_1 < i_2$ such that $r_1(i_1) = r_2(i_1)-1$, $r_1(i_2) = r_2(i_2) + 1$, for any $i \in \{1,\dots,i_1-1, i_1+1,\dots,i_2-1\}$, $r_1(i)=r_2(i)=0$ and for any $i \in \{i_2+1, \dots, m\}$, $r_1(i) = r_2(i)$, then we say that $\langle r_1, r_2\rangle$ can generate 2-type upper adjacency which is a vector whose component i is $\max(r_1(i), r_2(i))$ for any $i=1,2,\dots,m$, and we use $p_2(r_1,r_2)$ to denote the **2-type upper adjacency**.

Definition 3.4 Suppose $R_{k-1} \subseteq G_{k-1}$, $2 \le k \le \sum_{i=1}^{m} n_i$. Let

$GP_1(R_{k-1}) \overset{\Delta}{=} \{\alpha_{k-1} \mid \alpha_{k-1} \in R_{k-1}, \alpha_{k-1}$ can generate 1-type upper adjacency$\}$,

$GP_2(R_{k-1}) \overset{\Delta}{=} \{\langle r_1, r_2\rangle \mid r_1, r_2 \in R_{k-1}, \langle r_1, r_2\rangle$ can generate 2-type upper adjacency$\}$,

$P_1(R_{k-1}) \overset{\Delta}{=} \{ p_1(\alpha_{k-1}) \mid \alpha_{k-1} \in GP_1(R_{k-1})\}$,

$P_2(R_{k-1}) \overset{\Delta}{=} \{p_2(r_1, r_2) \mid \langle r_1, r_2\rangle \in GP_2(R_{k-1})\}$,

$P_1(R_{k-1})$ is called the set of all 1-type upper adjacencies of R_{k-1}, $P_2(R_{k-1})$ is called the set of all 2-type upper adjacencies of R_{k-1}.

Remark 3.1 Let D be a set of transactions and $I = \{i_1, i_2,\dots, i_m\}$ be the set of all items, then $P(I)$ is isomorphic with position lattice $\{0, 1\}^m$. If R_{k-1} is the set of all frequent (k-1)-itemsets, then $P_2(R_{k-1})$ is the set of all candidate k-itemsets before prune step and $P_c(R_{k-1})$ is the set of all candidate k-itemsets after prune step.

The following proposition gives the relations between several upper adjacencies.

Proposition 3.1 Let $R_{k-1} \subseteq G_{k-1}$, $2 \le k \le \sum_{i=1}^{m} n_i$, then $P_1(R_{k-1}) \subseteq P_c(R_{k-1}) \subseteq P_1(R_{k-1}) \cup P_2(R_{k-1}) \subseteq G_k$.

By Proposition 3.1, in order to obtain $P_c(R_{k-1})$, we can get $P_1(R_{k-1}) \cup P_2(R_{k-1})$ firstly, and then delete the elements which do not belong to $P_c(R_{k-1})$ from $P_2(R_{k-1})$.

Algorithm 3.1 GenCUA, finding all closed upper adjacencies

Input: a nonempty subset R, sorted by lexicographic order, of G_{k-1}, $1 \le k \le \sum_{i=1}^{m} n_i$.

Output: $P_c(R)$, set of all closed upper adjacencies of R.

Method:

(1) if $R = \varnothing$ or $R = G_K$ then
(2) return $\varnothing$;
(3) if $R = G_0$ then
(4) return G_1;
(5) $PcR := \varnothing$;
(6) r_1 := the first element of R;
(7) repeat
(8) if $r_1 \in GP_1(R)$ then
(9) $PcR := PcR \cup \{p_1(r_1)\}$;
(10) if r_1 is not the last element of R then {
(11) r_2:= the next element of r_1;
(12) repeat
(13) if $\langle r_1, r_2\rangle \in GP_2(R)$ and $p_2(r_1, r_2) \in P_c(R)$ then
(14) $PcR := PcR \cup \{p_2(r_1, r_2)\}$;
(15) next r_2;
(16) until r_2 moves out of R;
(17) };
(18) next r_1;
(19) until r_1 moves out of R;
(20) return PcR;

Definition 3.5 Suppose S is a lower segment of $\mathrm{Pos}(n_1,n_2,\ldots,n_m)$. for any $k(0 \le k \le \sum_{i=1}^{m} n_i)$, let $S_k \triangleq S \cap G_k$, then S_k is called the ***k*-grade** or ***k*-layer** of S.

Proposition 3.2 Let S be a lower segment of $\mathrm{Pos}(n_1,n_2,\ldots,n_m)$, $2 \le k \le \sum_{i=1}^{m} n_i$, $P_c(S_{k-1}) \subseteq R_k \subseteq G_k$, then $S_k = S \cap R_k$, especially, $S_k = S \cap P_c(S_{k-1})$.

4. Finding all elements of a lower segment

By Proposition 3.2, we can obtain all elements of a lower segment S by the following steps: generating $P_c(S_{k-1})$ by (k-1)-layer S_{k-1}; filtering out all the elements that do not belong to S and thus getting k-layer S_k from $P_c(S_{k-1})$; if S_k is not empty then join S_k to S. k increase (initial value is 1) until S_k is empty.

A lower segment S usually represents itself as a lower segment property P in the following definition.

Definition 4.1 Let P be a property of element in $\mathrm{Pos}(n_1,n_2,\ldots,n_m)$. If

$$S(P) \triangleq \{v \in \mathrm{Pos}(n_1,n_2,\ldots,n_m) | v \text{ has property } P\}$$

is a lower segment of $\mathrm{Pos}(n_1,n_2,\ldots,n_m)$, then we call P a **lower segment property** on $\mathrm{Pos}(n_1,n_2,\ldots,n_m)$.

Now we give the generalized Apriori algorithm for finding all elements of $S(P)$:

Algorithm 4.1 GenApriori, finding all elements of a lower segment

Input: natural numbers $n_1, n_2, \ldots, n_m$ and a lower segment property P on $\mathrm{Pos}(n_1,n_2,\ldots,n_m)$.

Output: all elements of $S(P)$.

Method:

(1) $S := \varnothing$;
(2) Sk := SubsetWithProperty($\{\langle 0, 0, \ldots, 0\rangle\}$);
(3) while $Sk \ne \varnothing$ do {
(4) $S := S \cup Sk$;
(5) $PcSk$:= GenCUA(Sk);
(6) $nextSk$:= SubsetWithProperty($PcSk$);
(7) $Sk := nextSk$;
(8) };
(9) return S;

procedure SubsetWithProperty($PcSk$)
(1) next $Sk := \varnothing$;
(2) for each $v \in PcSk$ do {
(3) if v has property P then
(4) $nextSk := nextSk \cup \{v\}$;
(5) };
(6) return $nextSk$;

The element v in step (2) of procedure SubsetWithProperty($PcSk$) is called a **test point**.

Example 4.1 Suppose P is the following lower segment property of Pos(4,3,4):

$$xyz \le 312 \text{ or } 231 \text{ or } 122,$$

then the variables S, Sk, $PcSk$; $nextSk$ change as follows:

$S = \varnothing$;
$Sk = G_0 \ne \varnothing$; $S = G_0$; $PcSk = G_1$; $nextSk = G_1$;
$Sk = G_1 \ne \varnothing$; $S = G_0 \cup G_1$; $PcSk = G_2$; $nextSk = G_2$;
...
$Sk = \{231, 312\} \ne \varnothing$;
$S = G_0 \cup G_1 \cup G_2 \cup \{012, 021, 030, 102,111, 120, 201, 210, 300\} \cup \{022, 031, 112, 121, 130, 202,211, 220, 301, 310\} \cup \{122, 131, 212, 221, 230, 302, 311\} \cup \{231, 312\}$;
$PcSk = \varnothing$; $nextSk = \varnothing$; $Sk = \varnothing$;

thus, the Algorithm 4.1 is over. The number of test points is 43. The number of elements in the lower segment is 38.

Remark 4.1 Let D be a set of transactions and $I = \{i_1, i_2, \ldots, i_m\}$ be the set of all items. If for any i, $n_i = 1$ and "v has property P " means "v is frequent itemset", then $\mathrm{Pos}(n_1,n_2,\ldots,n_m) \cong P(I)$, the algorithm GenApriori degenerates to R. Agrawal's Apriori algorithm (Sk is

position lattice	maximal elements of lower segment	the number of elements in lower segment	the number of test points	runtime (unit: second)
Pos(5,30,6)	⟨0,21,4⟩	110	110	0.906
Pos(5,30,6)	⟨2,9,3⟩	120	120	0.922
Pos(5,30,6)	⟨4,0,5⟩	30	30	0.250
Pos(5,30,6)	⟨0,21,4⟩, ⟨2,9,3⟩	190	190	1.625
Pos(5,30,6)	⟨0,21,4⟩,⟨2,9,3⟩,⟨4,0,5⟩	207	208	1.719
Pos(10,10,10,10,10)	⟨0,0,0,3,0⟩	4	5	0.046
Pos(10,10,10,10,10)	⟨2,2,2,2,0⟩	81	81	0.625
Pos(43,31,42,43,31)	⟨2,2,2,2,0⟩	81	81	0.625
Pos(43,31,42,43,31)	⟨2,2,2,2,1⟩	162	162	1.547
Pos(10,10,10,10,10,10,10)	⟨0,0,0,2,2,2,2⟩	81	81	0.735
Pos(10,10,10,10,10,10,10)	⟨2,2,2,2,0,0,0⟩	81	81	0.703
Pos(10,10,10,10,10,10,10)	⟨2,2,2,2,2,2,2⟩	2187	2187	84.219
Pos(4,3,4,5,5,5,5,5,5,5)	⟨0,2,0,4,0,3,0,0,0,0⟩	60	60	0.235

Table 1. some experimental results

L_{k-1}, the set of all frequent (k-1)-itemsets, and *PcSk* is the set of all candidate k-itemsets after prune step).

Remark 4.2 In Algorithm 4.1, the method for finding S_k is to filter out all the elements that do not belong to S from $P_c(S_{k-1})$. By Proposition 3.2 and Proposition 3.1, we know that $P_c(S_{k-1})$ can be replaced with $P_1(S_{k-1}) \cup P_2(S_{k-1})$. If doing so, the judgment that whether the elements in $P_2(S_{k-1})$ belong to $P_c(S_{k-1})$ is omitted. But we have to do additional work to judge the elements in $P_2(S_{k-1}) \setminus P_c(S_{k-1})$ do not have property P.

5. Experimental analysis

Some experimental results about Algorithm 4.1 are presented in Table 1 where the lower segments are constituted by the method in Example 4.1. By experimental results analysis, we get a general rule: The number of the test points is always greater than that of elements in the lower segment. Most of the test points are in the lower segment. The more the test points are, the longer the runtime is. The number of the test points has poor relation with the component upper bounds of position lattice. If both the number of nonzero components of maximal elements and the layers of maximal elements increase, the number of test points also increase. For the same number of nonzero components of maximal elements, the higher the layers of maximal elements are, the more the test points are. For the same layers of the maximal elements, the more the nonzero components are, the more the test points are.

6. Conclusion

By discussing relations between two adjacent layers of a lower segment of a direct product of finite finite-chains, we got a generalized Apriori algorithm GenApriori which can find all elements, from lower layer to upper layer, of a lower segment property on the direct product. When the finite-chains contain only the least element 0 and the greatest element 1 and the lower segment property is "v is frequent itemset", the algorithm GenApriori degenerates to R.Agrawal's Apriori algorithm. The experimental results given in this paper show the success of our algorithm. Like Apriori algorithm, GenApriori algorithm is not efficient when the maximal elements of the lower segment are in upper layers. For some application requiring finding all elements in some layer or below some layer, the GenApriori algorithm can display its abilities.

Acknowledgements

The work was supported by the National Natural Science Foundation of P.R. China (No. 60474022) and the Henan Province Natural Science Foundation of P.R. China (No. 2000520025, G2002026). We gratefully acknowledge the valuable and helpful suggestions given by the anonymous reviewers.

References

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," *Proc. of the 20th Int. Conf. on Very Large Data Bases (VLDB'94)*, pp. 487-499, 1994.

[2] S. Burris and H. P. Sankappanavar, "A Course in Universal Algebra," *Springer-Verlag, NY*, 1981.

[3] Bernard Kolman, Robert C. Busby and Sharon Ross, "Discrete Mathematical Structures," *Prentice-Hall, Inc.*, 1996.