A Novel Incorporate Algorithm of Concept Lattice

Xi Gong
School of Software Engineering, University of Science and Technology Liaoning, Anshan, 114051, China
E-mail: askdjy05gx@163.com

Abstract With the expansion of the research field, the research object of some original seemingly unrelated properties have been studied together. At this time, the number of attribute in formal context has changed. For the increased attributes, we need to construct a new concept lattice. The existing incremental building algorithms of concept lattice need the original formal context as the basis, with single attribute or a set of attribute of the object to rebuild the concept lattice. They can’t effectively utilize these existing concept lattice that have not relation in attributes. Here, the paper presents one new algorithm for incorporating concept lattice based on the existed concept lattices. We can directly build the “together lattice” from bottom to top by direct product operation on the existed concept lattices and the mapping relation between the direct product lattice of two existed concept lattices and the “together lattice”. Formal contexts that attribute sets have no intersection are fit for this algorithm.

Keywords-component: formal context; concept lattice; incorporating; direct product

I. INTRODUCTION

Since the formal concept analysis theory was put forward by German Wille[1] professor in 1982, with 30 years’ development, the formal concept analysis and its core data structure concept lattice was proved to be an effective tool of knowledge discovery and data analysis. It has been widely used in the field of knowledge discovery, software engineering, information retrieval, information filtering etc[2-6]. How to improve the efficiency of constructing concept lattice is a hotspot problem. The existing algorithm to structure concept lattice can be divided into two categories: batch algorithm [7,8] and incremental algorithm [9-10]. Along with the information storage increase gradually, it often need a lot of formal context merge together to form larger formal context. The space and time complexity of concept lattice grows exponentially with the scale of the increase of the formal context. If we use the existing incremental algorithms to reconstruct the concept lattice of the new formal context, the method consumed a mass of memory and time. This algorithm directly utilizes two existing formal context corresponding to the original concept lattices to construct the total concept lattice that corresponding to the merged formal context and solves the problem of low utilization ratio of the original concept lattice.

II. BASIC CONCEPTS AND BASIC THEOREMS[11]

Definition 1 A formal context is composed of two sets O and A and a relation I between O and A. The elements of O are called the objects and the elements of A are called the attributes. In order to express that object o is in relation I with attribute a, we write (o,a)I or oAa.

Definition 2 Given a set of objects X⊂O and a set of attributes Y⊂A from the formal context (O,A,I), two operators can be defined as:

\[ f(X) = \{ a \in A | \exists x \in X, xIa \} \]

\[ g(Y) = \{ o \in O | \exists y \in Y, oIy \} \]

Definition 3 For x⊂O, y⊂A, a pair (x,y) such that f(x)=y, g(y)=x, is called a (formal) concept. For a concept (x,y) the set x is called the extent and the set y the intent of the concept.

Definition 4 If (x1,y1) and (x2,y2) are two concepts of formal context (O,A,I) with x1⊂x2, then (x2,y2) is called the super concept of (x1,y1), written as (x1,y1)<(x2,y2). If (x1,y1)≤(x2, y2), (x1,y1)<(x2,y2) then written as (x1,y1)<(x2,y2). If there is no (x3,y3) such that (x1,y1)<(x3,y3)<(x2,y2) then (x2,y2) is called as the father concept of (x1,y1) and (x1,y1) is called as the child concept of (x2,y2). With respect to this partial order, the set of all formal concepts forms a complete lattice written as L(O,A,I). L(O,A,I) is the called lattice of the formal context (O,A,I).

Definition 5 The direct product of two partial order sets (M1,≤) and (M2,≤) is defined as a partial order set (M1×M2,≤), the “×” means that (x1,y1)≤(x2,y2) if (x1≤x2, y1≤y2).

Theorem 1 concepts lattice L(O,A,I) is a complete lattice. The infimum and supremum are defined as:

\[ \land (A,B,T) = (\land T, f(\land A), f(\land B)) \]

\[ \lor (A,B,T) = (g(\lor B), f(\lor A)) \]

T is a index set.

III. BASIC PRINCIPLE OF THE ALGORITHM

In this paper, the original formal context is (O,A1,I1). The concept lattice expressed by L1. The additional attribute set in the formal context is A2. Its corresponding formal context is (O,A2,I2), the concept lattice expressed by L2. Lx=L1∪L2 is the direct product of L1 and L2. Lx is a complete lattice. (c1,c2) is the element of Lx. c1 is the concept of L1 as well as c2 is the concept of L2. The set expressed by Cx. The total concept lattice is expressed by L. C expresses the set of all elements of L. When A1∩A2=∅, this algorithm is applicable.
Definition 6 $\phi : C \to C_4$ is a mapping that associates a concept of the total lattice $L$ to a concept of the direct product lattice $L_x$.

$$\phi((x,y))=((g(\cap \gamma A_4),\gamma A_4),(g(\cap \gamma A_2),\gamma A_2))$$

Proposition 1 The mapping $\phi$ is an order isomorphism that associates $L$ to $L_x$ and injective.

Proof: Consider $(x_1,y_1),(x_2,y_2) \in C$ with $(x_3,y_3) \subseteq (x_2,y_2)$ then

$$\phi((x_1,y_1))=((g(\cap \gamma A_4),\gamma A_4),(g(\cap \gamma A_2),\gamma A_2))$$

$$\phi((x_2,y_2))=((g(\cap \gamma A_4),\gamma A_4),(g(\cap \gamma A_2),\gamma A_2))$$

For $(x_1,y_1),(x_2,y_2) \Rightarrow y_2 \subseteq y_1 \Rightarrow y_2 \cap \gamma A_4 \subseteq y_1 \cap \gamma A_4$, $y_2 \cap \gamma A_4 \subseteq y_1 \cap \gamma A_4 \Rightarrow (g(\gamma A_4),(y_1 \cap \gamma A_4)) \subseteq (g(\gamma A_4),(y_2 \cap \gamma A_4))$ and $(g(\gamma A_2),(y_1 \cap \gamma A_2)) \subseteq (g(\gamma A_2),(y_2 \cap \gamma A_2))$ then $\phi((x_1,y_1)) \leq \phi((x_2,y_2))$. Obviously, the mapping $\phi$ is order-preserving.

Consider $\phi((x_1,y_1)) \neq \phi((x_2,y_2))$ with $(x_1,y_1),(x_2,y_2) \in C$ then $(g(\gamma A_4),(y_1 \cap \gamma A_4)) \neq (g(\gamma A_4),(y_2 \cap \gamma A_4)) \Rightarrow y_2 \neq y_1$. So, $\phi$ is an order isomorphism.

Proposition 2 The mapping $\phi$ is supremum-preserving.

Proof: Let $X=(x_1,y_1) \cap T$ be an index set, $\forall \in T$. $(x_1,y_1)$ is a concept of the total lattice $L \subseteq C$, so $X$ is a subset of $C$, then

$$\forall X=((g(\cap \gamma A_4),\gamma A_4),(g(\cap \gamma A_2),\gamma A_2))$$

$$\phi(X)=\phi((g(\gamma A_4),(\gamma A_4)),(g(\gamma A_2),(\gamma A_2)))$$

So properties 2 is supremum-preserving.

Definition 7 $\psi : C_4 \to C$ is a mapping that associates a concept of the direct product lattice $L_x$ to a concept of the total lattice $L$.

$$\psi((x_1,y_1),(x_2,y_2))=(x_1 \cap x_2, f(x_1 \cap x_2))$$

Proposition 3 The mapping $\psi$ is order-preserving and infimum-preserving.

Proof: If $((x_1,y_1),(x_2,y_2)),((x_3,y_3),(x_4,y_4)) \in C_4$ and $((x_1,y_1),(x_2,y_2)) \subseteq ((x_3,y_3),(x_4,y_4))$ then

$$\psi((x_1,y_1),(x_2,y_2))=((x_1 \cap x_2, f(x_1 \cap x_2))$$

$$\psi((x_3,y_3),(x_4,y_4))=((x_3 \cap x_4, f(x_3 \cap x_4))$$

So properties 2 is order-preserving.

Next, we proof the mapping $\psi$ is infimum-preserving.

Let $X=((x_1,y_1),(x_2,y_2))$ is an index set, $\forall \in T$. $(x_1,y_1)$ is a concept of covers in direct product lattice $L_x$. $(x_1,y_1) \in T$. So $Y$ is a concept of direct product lattice $L_x$.

$$\psi(Y)=\psi(((x_1,y_1),(x_2,y_2)))$$

$$=\psi((x_1 \cap x_2, f(x_1 \cap x_2)))$$

So properties 2 is infimum-preserving.

According to the above definition, theorem and proposition, may safely draw the following conclusion.The total lattice $L$ and the image on $L_x$ that are produced by mapping $\phi$ are isomorphic. With this conclusion, to produce the total lattice $L$ by one sublattice of the direct product lattice $L_x$ whom satisfied some condition become possible.

Properties 1 $\forall \in C$, $\phi(c)=\min(\psi^{-1}(c))$

Proof: Let $((x_1,y_1),(x_2,y_2))=\min(\psi^{-1}(c)) \subseteq C$ and any $((x_1,y_1),(x_2,y_2)) \subseteq \psi^{-1}(c)$ such that $((x_1,y_1),(x_2,y_2)) \subseteq ((c_1,t_1),(c_2,t_2))$.

$\exists (c_1,t_1),(c_2,t_2) \subseteq c$ such that $\forall c$.

$\forall (c_1,t_1),(c_2,t_2)$, $\subseteq (c_1,t_1),(c_2,t_2) \subseteq c$.

So properties 1 is tenable.

Definition 8 $\psi' : C \to C$ is a mapping that associates a concept of the total lattice $L$ to a concept of covers in direct product lattice $L_x$.

$$\psi'((x_1,y_1),(x_2,y_2))=((x_1 \cap x_2, f(x_1 \cap x_2))$$

$\psi'((x_1,y_1),(x_2,y_2))=((x_1 \cap x_2, f(x_1 \cap x_2))$.

$\psi'((x_1,y_1),(x_2,y_2))=((x_1 \cap x_2, f(x_1 \cap x_2))$.

So properties 2 is tenable.

Definition 9 $\psi' : C \to C$ is a mapping that associates a concept of total lattice $L$ to a concept of covers in direct product lattice $L_x$.

$\psi'((x_1,y_1),(x_2,y_2))=((x_1 \cap x_2, f(x_1 \cap x_2))$.

$\psi'((x_1,y_1),(x_2,y_2))=((x_1 \cap x_2, f(x_1 \cap x_2))$.

So properties 2 is tenable.

$\psi'((x_1,y_1),(x_2,y_2))=((x_1 \cap x_2, f(x_1 \cap x_2))$.

$\psi'((x_1,y_1),(x_2,y_2))=((x_1 \cap x_2, f(x_1 \cap x_2))$.

So properties 2 is tenable.

$\psi'((x_1,y_1),(x_2,y_2))=((x_1 \cap x_2, f(x_1 \cap x_2))$.

$\psi'((x_1,y_1),(x_2,y_2))=((x_1 \cap x_2, f(x_1 \cap x_2))$.

So properties 2 is tenable.

$\psi'((x_1,y_1),(x_2,y_2))=((x_1 \cap x_2, f(x_1 \cap x_2))$.

$\psi'((x_1,y_1),(x_2,y_2))=((x_1 \cap x_2, f(x_1 \cap x_2))$.

So properties 2 is tenable.

$\psi'((x_1,y_1),(x_2,y_2))=((x_1 \cap x_2, f(x_1 \cap x_2))$.

$\psi'((x_1,y_1),(x_2,y_2))=((x_1 \cap x_2, f(x_1 \cap x_2))$.

So properties 2 is tenable.

$\psi'((x_1,y_1),(x_2,y_2))=((x_1 \cap x_2, f(x_1 \cap x_2))$.
concepts by mapping \( \psi \). The maximum of them form the set \( \text{children}(c) \).

Definition 9 \( \text{children}(c)=\max(\{\psi(\tau)|\tau \in \text{children}(\varphi(c))\}) \)

If we want to build the total concept lattice with the original lattice and the additional lattice there are three questions need process.

(1)To find the update concepts from the direct product lattice that can produce new concepts.
(2)To compute the extent and the intent of these concepts.
(3)To define the father-son relation between these concepts.

(1)(2)(3) are the main steps of this algorithm and the above properties form the theory foundation. This algorithm can produce the total concept lattice only through the direct product operation on the original lattice and the additional lattice.

Input: direct product lattice \( L_x \), the result of direct product operation on The original lattice \( L_1 \) and the additional lattice \( L_2 \).

Output: \( L \), the total concept lattice

Procedure \( \text{Constret–lattice} \) (Input \( L_x \) , Output \( L \))

Begin
\( L \leftarrow \emptyset \)
For each \((c_i,c_j)\) in \( C_1 \times C_2 \) do
\( E \leftarrow \text{Extent}(c_i) \cap \text{Extent}(c_j) \)
\( \text{Judgevalue} \leftarrow \psi(\text{children}(c_i,c_j)) \)
// If this concept is an update concept
If \( \text{Findnew}(E, \text{Judgevalue}) \) then
// produce a new concept c. Its extent is E and intent is the union of \( c_i \)'s intent and \( c_j \)'s intent.
\( c \leftarrow (E, \text{Intent}(c_i) \cap \text{Intent}(c_j)) \)
For each \( \tau \) in \( \max(\text{Judgevalue}) \) do
\( \text{Link}(c, \tau) \) // \( \tau \) is \( c \)'s child concept. Link \( \tau \) to \( c \).
\( L \leftarrow L \cup \{c\} \) // Add the new concept \( c \) to the \( L \).
End
// To judge whether \((c_i,c_j)\) is updated concept. If the return value is true then \((c_i,c_j)\) is updated concept.

Produce \( \text{Findnew}(E, \text{Judgevalue}) \)

Begin
\( \text{flag} \leftarrow \text{false} \) // signal variable
For each \( \tau \) in \( \text{Judgevalue} \) do
If \( \text{Extent}(\tau) \neq E \) then
\( \text{flags} \leftarrow \text{true} \)
Else begin
\( \text{flag} \leftarrow \text{false} \)
// If \( \text{Extent}(\tau) = E \) then break the circulation and return the flag’s value.
return \( \text{flag} \)
end
return \( \text{flag} \)
End

// the algorithm get the child concepts of \((c_i,c_j)\)

Procedure \( \text{children}(c_i,c_j) \)

Begin
\( \text{child} \leftarrow \emptyset \)
For each \((c_m,c_n)\) in \( C_1 \times C_2 \) do
If \((c_m,c_n)\) and has no \((c_k,c_p)\) such that \((c_m,c_n) \prec (c_k,c_p) \prec (c_i,c_j)\) then
// \((c_m,c_n)\) is child concept of \((c_i,c_j)\)
\( \text{child} \leftarrow \text{child} \cup \{(c_m,c_n)\} \)
\( \text{children} \leftarrow \text{child} \)
End

IV. EXAMPLE

Here, we use a specific example to demonstrate the algorithm. The original formal context \((O,A_1,I_1)\). The additional formal context \((O,A_2,I_2)\). Let \( O = \{1,2\} \), \( A_1 = \{a,b,c\} \), \( A_2 = \{d,e\} \). These formal contexts and their concept lattices are shown as Figure 1. The total formal context \((O,A,I)\) is shown as Figure 2.

The lattice generated by direct product operation on two of the original formal context concept lattice is shown on the left in Figure 3.

Using the above algorithm on this lattice, firstly find the update concepts, secondly according to the properties 2 calculated the concept \( C \) of the combined total lattice. Finally, according to the definition 9 found child nodes of concept \( c \) and connecting them. It constructed a combined total concept lattice (such as the right side of Figure 3 shows).

![Figure 1](image1.png)

(a) The original formal context and its concept lattice

![Figure 2](image2.png)

(b) The additional formal context and its concept lattice

Figure 1. The original formal context, the additional formal context and their corresponding concept lattice
In order to verify the correctness of the algorithm, we compared this algorithm and the algorithm that produces concept lattice by formal context with experiments. The latter’s input is the total formal context and the output is the total concept lattice. The program use VB6.0 as programming language. Experiment data use random data. The number of objects is set to 30. The number of attributes is changing, from the beginning of 20, incrementing by 20 until 200. Each object has the same number of attributes. Context concentration is 40%. The experimental results prove the validity of the algorithm.

Algorithm features: Some of the existing incremental construction algorithms of concept lattice, their core idea is with a concept lattice as the basis, then every increase a attribute to traverse the concept lattice one time[12,13]. According to the additional attributes to determine concept category, and carries on the corresponding operation to form the new concept lattice until you add all attributes to generate the final result.

This algorithm provides a new method with the direct product for constructing concept lattice. Firstly, we produce the direct product lattice. Using the original concept lattice and direct product lattice isomorphic relation, can easily determine the father-son relationship of concepts in the direct product concept lattice. Concepts are only divided into two categories: update and non-update. The operation is simple.

The method only needs to traverse the direct product lattice one time. In the traverse, it adds all additional attributes and generates the concept lattice from the bottom to the top. It does not traverse and operate the concept lattice when there is a new attribute is added like the existing incremental algorithms. So, it is more efficient than the incremental algorithm.

V. CONCLUSION

Concept lattice generation problem is the key steps of knowledge extracting from database. To overcome the shortcomings of the present incremental building algorithms of concept lattice, we propose a new method based on direct product. It is applicable nicely when a large number attributes were added to database or more than two database were merged. The experiment proved that the algorithm is correct, but the application scope is limited. The next improvements include reducing the limitations of the algorithm, and to expand the scope of application of the algorithm, to reduce the search range of the update concept, as well as to improve the association rules extraction algorithm in data mining.

REFERENCES