A FAST ALGORITHM FOR BUILDING CONCEPT LATTICE

SHENG-YONG QIAO, SHUO-PIN WEN, CAI-YUN CHEN, ZHI-GUO LI

Center for Combinatorics, LPMC, Nankai University, Tianjin, 300071, P. R. China
E-MAIL: qiaoshyy@eyou.com

Abstract

The concept (or Galois) lattice produced from a binary relation has been proved useful for many applications. But building concept lattice is difficult. Reference [5] gives a fast algorithm for building lattice, but concept lattice have some special characters. This article presents a fast algorithm for building concept lattice and corresponding graph, and also gives the time complexity of this algorithm. Database updating is always necessary. In our paper, we also give an algorithm for modify lattice when dada update happened.

Keywords
Algorithm, Concept lattice, Basis

1. Introduction

Building the concept lattice of a binary relation has many important applications. Wille (1982) proposed to consider each element in the lattice as concept and the corresponding graph (Hasse diagram) as the generalization/specialization relationship between concepts. From this ideal, the lattices represents a concept hierarchy. Concept hierarchy has been shown to have many advantages in the field of knowledge discovery from large database. Many algorithms have been proposed for generating the concept lattice from a binary relation [1,2,4]. But very little is known about the complexity of these algorithms. Lhouari Nourine et al have proposed a fast algorithm for building lattice[5]. In this paper, we will present an improved version and modify it to building concept lattice. The algorithms we give generate both the concept and Hasse diagram .We also give the complexity of the algorithms.

In our algorithm, we compute the basis of the context above all; From the basis of the context, we derived all the concepts of the context; Taking advantage of theorem 4 in our paper we can easily compute the cover relation of the concepts of the context.

Database update is always necessary. It is important in many applications to be able to incrementally add some new objects, by modifying the lattice without having to regenerate it from scratch. In this paper we also give an efficient algorithm to incrementally add some new objects into the concept lattice. When the context include a lot of data, building concept lattice is very difficult even impracticable. Our algorithm sometime can effectively solve this problem.

The following section recalls basic definitions related to the concept of Galois lattice. Section 3 describes our algorithm for building lattice. Section 4 presents an algorithm to add new data to the existed lattice.

2. Basic Notions

In this section we will recall necessary basic notions used in our paper. The detail description about concept lattice can be found in [2].

Definition 1. A *formal context* **K :=** (G,M,I) consists of two sets G and M and a relation I between G and M . The elements of G are called the *objects* and the elements of M are called the *attributes* of the context. In order to express that *an object g is in a relation I with an attribute m*, we write *gIm* or (g, m) $\in$ I, and read it as "the abject g has the attribute m".

Example 1. A small context can be easily represented by a *cross table*, such as Figure 2.1. It represents a formal context **K** = (G,M,I) ,where G={1,2,3,4,5} and M ={a,b,c,d},and a cross in row g $\in$ G and column m $\in$ M means that the object g has the attribute m.

	a	b	c	D
1	×	×		
2			×	×
3	×		×	
4	×	×	×	
5				×

Figure 2.1

Definition 2. For a set A $\subseteq$ G of objects we define A' := {m ∈ M | g I m for all g ∈ A}(the set of attributes common to the objects in A).Correspondingly, for a set B of attributes we define B' := {g ∈ G | g I m for all m ∈ B}.(the set of objects which have all attributes in B).

Definition3. A *formal concept* of the context (G,M,I) is a pair (A,B) with $A \subseteq G$, $B \subseteq M$, A'=B and B' = A. We call A the *extent* and B the *intent* of concept (A,B). ***B****(G,M,I)* denotes the set of all concept of the context (G,M,I) .

A partial order relation can be defined on the set **B**(G,M,I) of all concept of the context (G,M,I), Given H_1 = (X' ,X) ∈ **B**(G,M,I) and H_2 = (Y' ,Y) ∈ **B**(G,M,I), define $H_1 < H_2 \Leftrightarrow X \subseteq Y$, and the precedent order means H_1 is parent of H_2 or direct generalization in the lattice. In fact, there is a dual relationship between X' and X in the lattice, i.e., $X \subseteq Y \Leftrightarrow Y' \subseteq X'$. So, concept lattice is two lattice connected together in essence. The Hasse diagram of the lattice can be generated by use of the partial order relation. If $H_1 < H_2$ and there is no other elements such that $H_1 < H_3 < H_2$, there is an edge from H_1 to H_2. It reveals the generalization/specialization relationship between the concepts and could be used as an efficient tool for data mining and knowledge acquisition.

Example 2. The context in Example 1 has 9 concepts. The line diagram in Figure 2.2 represents the concept lattice of the context.

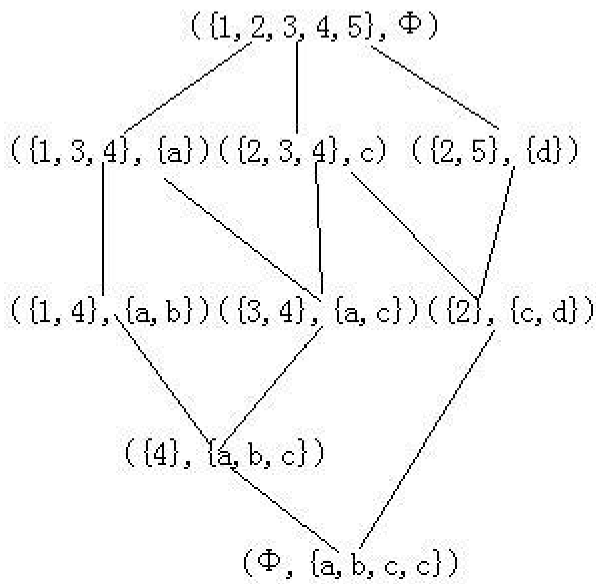

Figure 2.2 Concept lattice for context of Figure 2.1

3. A fast algorithm for constructing the concept lattice

Before introducing the algorithm, we give some definitions and notations.

Let **K** =(G,M,I) be a formal context., for an object g ∈ G , we write g' instead of {g}' for the *abject intent {m ∈ M | g I m }* of the object *g*. Correspondingly, *m': ={ g ∈G | g I m}* is the *attribute extent* of the attribute *m*. A *Basis* B is the set of all attribute extent of **K,** i.e.,

$$\mathrm{B} = \{m' \mid m \in M\}.$$

We denote by F_B the family generated by joint form the basis B, i.e.,

$$F_{\mathrm{B}} = \{\bigcap_{m' \in I} m' \mid I \in 2^{\mathrm{B}}\}.$$

For each $F \in F_{\mathrm{B}}$, we denote by *γ(F)* the subset of M, such that for each element m in it , F is included in m' , i.e.

$$\gamma(F) = \{m \in M \mid F \subseteq m'\}$$

Example 3 In the Context of Figure 1.1 , *B* = {a'={134}, b'={1,4}, c'={234}, d'={25}}, F_B = {{12345}, {134}, {14}, {234}, {34},{25}, {2}, {4}, Φ } and { γ(F) | $F \in F_{\mathrm{B}}$ } = {Φ,{a},{ab},{c},{d},{ac},{cd},{abc},{abcd}}

The following theorem can be obviously derived from the above definition.

Theorem1 Let **K=** (G, M, I) be a formal context, for each $F \in F_{\mathrm{B}}$, (F, γ(F)) is a concept of **K,** and **B** ≡ {(F, γ(F))| $F \in F_{\mathrm{B}}$ } = **B**(G,M,I).

Now our problem is constructing the concept lattice from the given context **K**=(G, M, I). We use a algorithm with O((|G|+|M|)*|M|*|F_{B}|) time complexity to do this. Our algorithm can be complete by the following three steps:

(1) Compute the basis of the context B;
(2) Generate the family **B =** {(F, γ(F))| $F \in F_{\mathrm{B}}$ };
(3) Construct the concept lattice from **B.**

The next third subsections will introduce the algorithm for each step, respectively.

3.1 Compute the basis *B* of the context K

From the definition of the basis, we can see the cardinal of *B* is equal to the cardinal of set *M*, i.e., |*B* | = | *M* |.

Algorithm1. Basis *B* .
Input: the context K=(G,M,I)
Output: the basis *B* of the context.
Begin
\# Initialize *B*
Let |*B*| = |M|
for each m' ∈ *B do*
    m'= Φ
for each m' ∈ *B do*
    for each g ∈*G do*
        if gIm *then* m'= m' ∪ {g}

end

Clearly we have

Theorem 2 Algorithm 1 Computes the basis B of the context **K** in O(|G|*|M|) steps and with space in O(|M|).

Now let us show how use the basis B to generate the family **B** = {(F, γ (F))| F ∈F_B }.

3.2 Generate the family B = {(F, γ (F))| F ∈F_B }

By the following algorithm ,we generate all the concepts (F, γ (F)) of the given context **K** , that is to compute F_B and for each element F ∈F_B , γ (F) by the basis B with O((|G|+|M|)*|M|*| F_B |) time complexity.

Algorithm2 .Generate **B**(G,M,I) = {(F, γ (F))| F ∈ F_B }.

Input: A basis B

Output: **B**(G,M,I)

Begin

initialize F_B

Let F_B = {G}, γ (G) = Φ

for each m' ∈ *B do*

if m' = G *then* γ (G) = γ (G) ∪ {m}

for each m' ∈ *B do*

  for each F ∈F_B *do*

  begin

  F' = F ∩ m'

    if F' ∉ F_B *then*

      begin

        F_B = F_B ∪ F'

      end

        γ (F') = γ (F') ∪ {M}

  end

end

Similarly to References [5], we can get

Theorem 3 Algorithm 2 computes the family **B** = {(F, γ (F))| F ∈ F_B } generated by B in O((|G|+|M|)*|M|*| F_B |) steps, with space in O((|G|+|M|)*|M|) .

The next subsection we will show how to use the concepts set **B** to construct the concept lattice.

3.3 Construct the concept lattice from B.

Let (F_B, , ⊂) be a partial order of the including relation of the sets .Let F' and F ∈ F_B with F' ⊂ F, we denote *D(F' F)* = γ (F')\ γ (F) and define the *exactly including relation* ≺ of F_B as follows :

$\forall F_1, F_2 \in (F_B, \subset)$ *,If* $F_1 \subset F_2$*,and not exist* $F \neq F_1, F_2$*,such that* $F_1 \subset F \subset F_2$ *,then we call* F_2 *exactly including* F_1 *and write* $F_1 \prec F_2$.

Example 3 For example 2 ,let F' = Φ ,F = {2}, γ (F') = {abcd}, γ (F)={cd}, then D(F',F)={ab}. Clearly F' ≺ *F, and we notice that F \ a' = F \ b' = {2}* , more generally, we have

Theorem 4 Let F' and F ∈ F_B with F' ⊂ F, then F' ≺ F ⇔ F\ m_1' = F \ m_2' for all m_1' ,m_2' ∈ D(F', F)

Proof Notice F' can be written as F' = F ∩ {m' | m' ∈ D(F', F)} then

=> for m_1' ,m_2' ∈ D(F', F), suppose F\ m_1' ⊄ F \ m_2' ,then we have F' = F ∩ {m' | m' ∈ D(F', F)} ⊂ F'' ≡ F ∩ B_1 ⊂ F, which is a contradiction with F' ≺ F, thus F\ m_1' ⊆ F \ m_2', Similarly we get F\ m_1' ⊇ F \ m_2'.

<= Suppose there exists F'', such that *F' ⊂ F'' ⊂ F then we have* γ *(F)* ⊂ γ *(F'')* ⊂ γ *(F').Since* γ (F'')\ γ (F) ⊆ γ (F')\ γ (F) = D(F',.F),then F'' = F ∩ {m' | m' ∈ D(F'', F)} =F'.

Clearly , we get the following corollary :

Corollary 1 Let F' and F ∈ with F' ⊂ F, then F' ≺ F ⇔ F' = F ∩ m' for all m' ∈ D(F', F).

Now we will introduce how to construct the concept lattices of given context **K** by **B.** Our task is that to each element F of F_B, we should find all the elements of F_B that exactly included in F, i.e. ∀ F ∈ F_B, find {F' ∈ F_B|F' ≺ F}. Clearly F' ∈ F_B is a candidate if F' ⊂ F and F' can be computed as F ∩ m' ,for some m' in B\ γ (F). Let us denote by S ={ F ∩ m'| m' ∈ B\ γ (F)}.Then our algorithm bases on the following theorem:

Theorem 5 F' ∈ S, F' ≺ F if and only if F' is exactly found |D(F',F)| times in S.

Proof From the definition of S, it is a direct consequence of Corollary 1.

In our algorithm, we compute the set S and maintain the number of occurrences of each element F' in S in a counter COUNT(F'). Then ,for each element F' of S, we check if the cardinal of D(F',F)is equal to COUNT(F'). If so, then F' ≺ F. In the Hasse diagram, there is a line from (F, γ (F)) to (F', γ (F')).

Algorithm 3. Construct the concept lattice from **B.**

Input: B

Output: the Hasse diagram of **B**(G,M,I)

Begin

#initialize

for each F ∈ F_B do

COUNT(F) = 0

for each F $\in F_B$ *do*
for each m' $\in B \setminus \gamma$ (F) *do*
begin
F' = F $\cap$ m';
COUNT(F')++;
if | γ (F') | = COUNT(F') + | γ (F) | *then*
Add a line from (F, γ (F)) to (F', γ (F')).
end
Reset COUNT;
End

Theorem 6 Algorithm 3 Constructs the concept lattice from **B** in O((|G|+|M|)*|M|*| F_B |) steps.

Now we have gotten a Hasse diagram from a context .It is clearly that the total complexity of our algorithm is O((|G|+|M|)*|M|*|F_B|) , It is a simple and efficient algorithm for building concept lattice.

In the next section, we will give an algorithm to add new data to the existed lattice, we will use algorithm 3 to realize the insertion of new concepts.

4. Add new data to the existed lattice (Incremental algorithm for building concept lattice)

Database update is always necessary. It is important in many applications to be able to incrementally add some new objects, by modifying the lattice without having to regenerate it from scratch. We know that the new objects is similar to the objects that have already existed in the concept lattice because the data belong to the same database. We suppose that concept lattice which construct from the new objects is alike to the existed concept lattice. Then a new problem comes up: Whether we can find a faster way to unite the two lattices. After study we find it is not easy to do this. But take advantage of the similarity of the two lattice, we give a more efficient algorithm.

The concepts set of the context will change when some new data is added to the context. The changing includes two categories. The first is only the extent of the concepts in the existed lattice have changed. The second is some new concept generated. For the first change, we only need modify the extent of the concepts in the existed lattice. This is easy to do. But for the second change, we need insert the new concepts to the existed lattice, this can be done using the incremental algorithm for building concept lattice in [1,4], but we can using algorithm 3 to realize it.

Now we give some notations using in the following algorithm:

B : the basis of the context before added new data
F_B : the set of the all the concepts generated by B
γ (F_B) : the set of the intent of the concept set F_B
γ (F_B) = { γ (m') | m' $\in B$}.
B' : the basis of the context after added new data
$F_{B'}$: the set of the all the concepts generated by B'

Algorithm 4 *add new data sets to the lattice*
Input: the basis *B', the primary lattice* ***L***
Output: *modified lattice*
begin
\# Initialize $F_{B'}$
Let $F_{B'}$ = {G}, γ (G) = Φ
for each m' $\in B'$ *do*
if m' = G *then* γ (G) = γ (G) $\cup$ {m}
for each m' $\in B'$ *do*
for each F $\in F_{B'}$ *do*
begin
F_0' = F $\cap$ m'
If $F_0' \notin F_{B'}$ *then*
$F_{B'} = F_{B'} \cup F_0'$
γ (F_0') = γ (F_0') $\cup$ {m}
if exits F' $\in F_B$ S.T. γ (F_0') = γ (F') *then*
F'= F' $\cup$ F_0'
else
insert concept (F_0', γ (F_0')) to the existed lattice ***L*** (*)
end if
end
end

Using this algorithm, we can add some new data to the existed lattice. From the algorithm we can see that only step (*) need to modify the structure of the lattice. We can use algorithm 3 to derive the relation of the new concepts and the concepts in the lattice. This can be done with O((|G|+|M|)*|M|*| $F_{B'}$ |) steps. It is more efficient compared with the incremental algorithm for building concept lattice in[1,4].

5. Conclusion

Concept hierarchy has been shown to have many advantages in the field of knowledge discovery from large database; Concept lattice is a convenient tool to data analysis and knowledge discovery. Therefore, Building concept lattice from context is an important problem of data mining. In our paper, we proposed a new fast algorithm to building concept lattice, and in the end of the paper, we give an efficient algorithm to add some data to the lattice. They will do much help to build lattice from big database.

6. Acknowledgments

This work was done under the auspices of the National “973” Project on Mathematical Mechanization and the National Science

Foundation of China.

Reference

[1] Godin,R,Missaoui,R,alaui,H. Incremental Concept Formation Algorithm Based on Galois (Concept) Lattice. Computational Itelligence,1995,11(2):246-267
[2] Bernhard Ganter,Rudolf wille. Formal Concept Analysis,1999,Springer-Verlag Berlin Heidelberg
[3] Xie Zhipeng Liu Zong-Tian A Fast Incremental Algorithm for Building Concept Lattice, Chinese J.Computer. 2002,25(5).
[4] Keyun Hu,Yuchang Lu and Chunyi shi Incremental Discovering Association Rules: A Concept Lattice Approach.
[5] Lhouari Nourine,Olivier Raynaud. A fast algorithm for building lattice Information Processing letters 71 (1999) 199-204