An Algorithm based on Concept-Matrix for Building Concept Lattice with Hasse

Sujing Wang
Jilin University
sujingwang@hotmail.com

Zhen Chen
Jilin University
chenzhen@jlu.edu.cn

Dongjing Wang
Nanjing University of IST
rpcwangdongjing@163.com

Abstract

As the core data structure of FCA (Formal Concept Analysis), concept lattice has been widely used in the field of machine learning and data mining. It is useful to study algorithms of building concept lattice in practical applications. There are several of algorithms of building concept lattice which have been developed. This paper presents an efficient algorithm named CMCG (Concept-Matrix based Concepts Generation) for building concept lattice and corresponding Hasse graph based on concept-matrix which is a novel notion.

Keywords: Concept lattice, rank of matrix, formal concept analysis

1. Introduction

Formal Concept Analysis (FCA) was developed by professor Wille in 1982 [1]. Concept lattice, the core data structure in Formal Concept Analysis, has been widely used in data mining and knowledge discovery. For example, Sahami [2] induces classification rules using a concept lattice. Another example is that Zaki [3] finds all closed itemsets by a concept lattice.

Every node of concept lattice is a formal concept consisting of extent and intent. Concept lattice embodies the relations between extension and intension among these concepts by Hasse graph. However, the efficiency of building concept lattice is most unsatisfactory. So people research it widely nowadays and propose any different algorithms which can be divided into two main categories, batch construction [4; 5] and incremental construction [6]. This paper presents an efficient algorithm named CMCG for building concept lattice and corresponding Hasse graph based on concept-matrix which is a novel notion.

2. Problem definition and related work

To make the paper self-contained, we introduce the basic notions of formal concept analysis [7].

Definition 1. A formal context is a triple: $K = (O, A, R)$, where $O$ and $A$ are two sets, and $R$ is a relation between $O$ and $A$. $O = \{o_1, ..., o_n\}$, each $o_i (i \leq n)$ is called an object. $A = \{a_1, ..., a_m\}$, each $a_j (j \leq m)$ is called an attribute.

In a formal context $K = (O, A, R)$, if $(o, a) \in R$, we say that the attribute $a$ is an attribute of the object $x$, or that $x$ verifies $a$. $(o, a) \in R$ is denoted by 1, and $(o, a) \notin R$ is denoted by 0. Thus, a formal context can be represented by a matrix only with 0 and 1. We say that the matrix is the context-matrix of $K$. [8] Table 1. represents a formal context. Figure 1. shows the matrix of the formal context represented in Table 1.

Table 1. A formal context

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Definition 2. Let $K = (O, A, R)$ be a formal context. We define a function $f(X)$ that produces the set of their common attributes for every set $X \subseteq O$ of objects to know which attributes from $A$ are common to these entire objects: $f(X) = \{y \in A | \forall x \in X, (x, y) \in R\}$

Dually, we define $Y$ for subset of attributes $Y \subseteq A$, $g(Y)$ denotes the set consisting of those objects in $O$ that have all the attributes from $A$: $g(Y) = \{x \in O | \forall y \in Y, (x, y) \in R\}$

These two functions are used to determine a formal concept.

Definition 3. Let $K = (O, A, R)$ be a formal context. A pair $(X, Y)$ is called a formal concept of $K$, for short, a...
concept, if and only if $X \subseteq O$, $Y \subseteq A$, $f(X) = X$ and $g(Y) = X$. $X$ is called extent, $Y$ is called intent.

$\begin{array}{cccccc}
 a & b & c & d & e & f \\
1 & 1 & 1 & 0 & 0 & 0 \\
2 & 1 & 1 & 0 & 1 & 0 \\
3 & 0 & 1 & 1 & 0 & 1 \\
4 & 1 & 0 & 0 & 1 & 0 \\
5 & 0 & 0 & 1 & 0 & 0 \\
6 & 0 & 0 & 0 & 1 & 0 \\
7 & 0 & 0 & 0 & 0 & 0 \\
\end{array}$

**Figure 1.** The matrix of the formal context represented in Table 1.

Definition 4. Let $K = (O, A, R)$ be a formal context. The set of all concepts of $K$ is denoted by $B(K)$, $C_1 = (X_1, Y_1)$ and $C_2 = (X_2, Y_2)$ are two concepts in $B(K)$. A partial ordering relation ($<$) is defined on $B(K)$ by: $C_1 < C_2 \iff X_1 \subset X_2$ or $C_1 < C_2 \iff Y_1 \supseteq Y_2$.

We say that $C_2$ is called a superconcept of $C_1$ and $C_1$ is called a subconcept of $C_2$, $B(K)$ and the partial ordering relation ($<$) form a complete lattice called the concept lattice of $K$ and denoted by $L(K)$.

The context in Table 1 has 16 concepts. The line diagram in Figure 2 represents the concept lattice of this context.

![Concept lattice for the context of Table 1](image)

**Figure 2.** Concept lattice for the context of Table 1.

Definition 5. Let $K = (O, A, R)$ be a formal context. $C_1$ and $C_2$ are two concepts in $B(K)$. If $C_1 < C_2$ and there is no concept $C_3$ in $B(K)$ fulfilling $C_1 < C_3 < C_2$, $C_1$ is called a lower neighbor of $C_2$ and $C_2$ is called an upper neighbor of $C_1$.

The set of all lower neighbors of a given concept is a subset of the set consisting of all subconcepts of it.

3. CMCG algorithm

3.1. Definitions and Theorem about Concept-Matrix

Definition 6. Let $K = (O, A, R)$ be a formal context. The concept-matrix of $C$ is the matrix consisting of these rows what are the corresponding rows of the each element $x$ in set $X$ in context-matrix of $K$.

$$
\begin{array}{cccccc}
 a & b & c & d & e & f \\
1 & 1 & 1 & 0 & 0 & 0 \\
2 & 1 & 1 & 0 & 1 & 0 \\
3 & 0 & 1 & 1 & 0 & 1 \\
4 & 1 & 0 & 0 & 1 & 0 \\
5 & 0 & 0 & 1 & 0 & 0 \\
6 & 0 & 0 & 0 & 1 & 0 \\
7 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
$$

**Figure 3.** The concept-matrix of concept (124.a)

Figure 3 represents the concept-matrix of concept (124,a).

Definition 7. Let $K = (O, A, R)$ be a formal context. $C = (X, Y)$ is a concept in $B(K)$. If the count of $1$s in the corresponding column of the attribute $y$ in the concept-matrix of $C$ is $t$, we say that the rank of attribute $y$ in concept-matrix of concept $C$ is $t$, denoted by $R_C(y) = t$. If $m = \max\{R_C(y) | y \in A, y \not\in Y\}$, we say that the rank of concept $C$ is $m$.

Property 1. The count of objects of subconcept of concept $C$ is equal or lesser than $m$.

Definition 8. Let $K = (O, A, R)$ be a formal context. $C = (X, Y)$ is a concept in $B(K)$. Given subset $Y_1 \subseteq A$. $g_C(Y_1)$ denotes the set consisting of those objects in $X$ that have all the attributes from $A$:

$$
g_C(Y_1) = \{x \in X | \forall y \in Y_1. (x, y) \in R\}
$$

Theorem 1. Let $K = (O, A, R)$ be a formal context. $C = (X, Y)$ is a concept in $B(K)$. The rank of concept $C$ is $m$. For $\forall y \in Y | R_C(y) = m, y \in A$,

$$
C_1 = \left( g_C(Y), f(g_C(y)) \right)
$$

Then $C_1$ is a lower neighbor of $C$.

Proof. By Definition 8, we have $|g_C(Y)| = m$. Suppose there exist $C_2 = (X_2, Y_2)$, where $C_1 < C_2$. We can obtain $m = |g_C(Y)| < |X_2| < |X|$. By Property 1, we have $|X_2| \leq m$. This result contradicts with $m < |X_2|$. Then $C_1$ is a lower neighbor of $C$.

Theorem 2. Let $K = (O, A, R)$ be a formal context. $C = (X, Y)$ is a concept in $B(K)$. The rank of concept $C$ is $m$. $C_1 = \left( g_C(Y_1), f(g_C(Y_1)) \right)$ is a subconcept of $C$, where $Y_1 \subseteq A$ and $R_C(Y_1) = m_1 > 0$. For $\forall \omega \subseteq C_2 = (X_2, Y_2)$, $C_2 \subseteq C, m_1 < |Y_2|$ , there exist $g_C(Y_1) \subseteq X_2$. Then $C_1$ is a lower neighbor of $C$.

Proof. Suppose there exist $C_2 = (X_2, Y_2)$, where $C_1 < C_2 < C$. We have $m_1 = |g_C(Y_1) \cap X_2| < |X_2| < |X|$ . it implies that $C_2 \subseteq (C_2 \subseteq (X_2, Y_2) \subseteq C, m_1 < |X_2|$ , and $g_C(Y_1) \subseteq X_3$. On the other hand $C_1 < C_3 < C$ implies that $g_C(Y_1) \subseteq X_3$. This result contradicts with $g_C(Y_1) \subseteq X_3$. Then $C_1$ is a lower neighbor of $C$.

3.2. CGCM algorithm

Now we have discussed the principle for generating concepts based on concept-matrix. In this section, the corresponding algorithm is addressed. The main procedure of the algorithm will generate the largest concept $(O, \emptyset)$ and put it into a graph. Then all subnodes of each leaf-node in...
the graph is generated by calling function Subnode() and added into the graph. The pseudo codes of the main procedure are given below.

### Algorithm CMCG

**Input:** Formal Concept \( K = (O, A, R) \)

**Output:** all concepts on \( K \) and the corresponding Hasse graph \( G \)

1: Initialize a graph \( G \);
2: Generate concept \((O, \emptyset)\), and add vertex \((O, \emptyset)\) into \( G \);
3: \( S \leftarrow \) all leafnodes in \( G \);
4: if \(|S| = 1 \) and \((\emptyset, A)\) in \( S \) then return;
5: for each \( C \) in \( S \)
6: \( S_1 = \text{Subnodes}(C) \);
7: for each \( C_i \) in \( S_1 \)
8: \( \text{add vertex } C_i \text{ into } G; \)
9: \( \text{add edge } (C, C_i) \text{ into } G; \)
10: \( \text{loop} \)
11: \( \text{loop} \)

It is necessary to search if node \( C_i \) is in graph \( G \) in Line 8, then add vertex into graph \( G \). The runtime of this search increases quadratically with the size of the concept lattices. For this purpose all concepts are stored in a search tree in Lattice Algorithm. For a search tree, with the increase of the number of the nodes, the runtime of this search is more. Besides, if a search tree is substituted for an AVL tree, a sequence of rotations is performed to maintain balance of AVL tree when adding a node, which is time-consuming.

In CMCG Algorithm, all concepts are stored in a Trie tree. A concept can be identified with its intent or extent, so we let the intent of a concept as key to identify this concept. Because of a given formal context \( K = (O, A, R) \), there is usually \(|O| > |A|\). The intent of a concept is expressed as a fix-length string consisting of 0 or 1. The length of this string is\(|A|\). The intent of a concept is stored in a Trie tree whose degree is 2 and depth is \(|A|\). For example, it’s intent \( c_f \) of concept \((5, cf)\) is expressed as 00100100.

Its advantages are followed: i) the runtime of searching for a node is constant which is independent of the number of the nodes. ii) Because it takes key words separately to the nodes of Tire tree for storage, it needs less space.

Then we introduce the idea of the function Subnodes(). The set of all lower neighbors of concept \( C \) is obtained by computing the ranks of every attribute \( a \in A \) in concept-matrix of concept \( C \). The set is denoted by \( \text{subnodes} \).

Let’s examine how to get all lower neighbors of a given concept \( C \) using the following example.

For the formal context shown in Table 1, we get all lower neighbors of concept \((124, a)\). Let \( m = 2 \). It can be drawn form Figure 3 that \( R_C(b) = R_C(d) = R_C(g) = 2 = \text{the rank of } C \). According to Theorem 1, a lower neighbor of \((124, a)\) is \((g_c(b), f(g_c(b))) = (12, ab)\). Since the corresponding column of \( d \) is same as one of \( g \), the attribute \( y \) in Theorem 1 can be considered as a set consisting of attributes which columns are same in order to avoid redundant computation.

So \((g_c(dg), f(g_c(dg))) = (24, adg)\) is a lower neighbor of \((124, a)\).

Then let \( m = m - 1 = 1 \) and \( R_C(c) = R_C(e) = 1 \). Since \( g_c(c) = \{1\} \subseteq \{12\}, \ (g_c(c), f(g_c(c))) \) is NOT a lower neighbor of \((124, a)\) according to Theorem 2. And for the same reason, \((g_c(e), f(g_c(e)))\) is NOT a lower neighbor of \((124, a)\). Therefore, all lower neighbors of \((124, a)\) are \((12, ab)\) and \((24, adg)\). The pseudo codes of the function Subnodes() are given below.

### Function Subnodes(C)

**Input:** Concept \( C = (X, Y) \)

**Output:** the set neighbors of all lower neighbors of Concept \( C \)

1: \( \text{subnodes} \leftarrow \emptyset; \)
2: if \(|X| = 1 \) then return \((\emptyset, A);\)
3: \( M \leftarrow \) the concept-matrix of concept \( C; \)
4: \( \text{compute the rank of the every attribute in } C; \)
5: \( m \leftarrow \text{the rank of } C; \)
6: if \( m = 1 \) then return \((\emptyset, A); \)
7: do while \( m > 0 \)
8: \( S \leftarrow \text{the set of the attributes which ranks equal to } m; \)
9: do while \( S \neq \emptyset \)
10: \( Y_i \leftarrow \text{the set consisting of a attribute } a \text{ from } S \text{ and these attributes from } S \text{ which corresponding columns are same as column of } a \text{ in } M; \)
11: \( S \leftarrow S - Y_i; \)
12: \( X_i \leftarrow g_C(Y_i); \)
13: \( Y_i \leftarrow Y \cup Y_i; \)
14: if \( \forall C_k \in (X_1, Y_1) \in \text{subnodes}, \) where \( X_1 \subset X \land X_2 \text{ then subnodes} \leftarrow \text{subnodes} \cup (X_i, Y_i); \)
15: \( \text{loop} \)
16: \( m \leftarrow m - 1; \)
17: \( \text{loop} \)
18: return \( \text{subnodes}; \)

### 4. Evaluation

We conducted a set of tests in which both variants of CMCG have been compared to the algorithm Lattice. The experiments were performed on a 1.7 GHz Intel processor with 1.0 GB main memory, running Linux 2.65 system. Both algorithms were implemented in C++.

The experiment used 2 groups of data where are randomly generated by Galicia(http://www.iro.umontreal.ca/~galicia/).

Group One corresponding context tables were \( 50 \times 20, 50 \times 25, 50 \times 30, \ldots, 50 \times 200 \) elements. Group Two corresponding context tables were \( 50 \times 20, 100 \times 20, 150 \times 20, \ldots, 3000 \times 20 \) elements. The context fill ratio which is the quotient of \(|R|\) and \(|O| \times |A| \) is 30%.
Figure 4. Running time of algorithms versus lattice size on Group One

Figure 5. Running time of functions which is to get all lower/supper neighbors of a given concept versus lattice size on Group One

Figure 6. Running time of algorithms versus lattice size on Group Two

Figure 7. Running time of functions which are to get all lower/supper neighbors of a given concept versus lattice size on Group Two

5. Conclusion

The theory of concept lattice is an effective tool for knowledge representation and knowledge discovery, and is applied to many fields. This paper presents an algorithm of generating concept lattice with Hasse based on concept-matrix. By experiment the effectiveness of CMCG Algorithm was proved contrasting to Lattice Algorithm.

6. References