Expert Systems with Applications 42 (2015) 4474–4481

Contents lists available at ScienceDirect

Expert Systems with Applications

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

A fast incremental algorithm for constructing concept lattices

Ligeng Zou, Zuping Zhang *, Jun Long

School of Information Science and Engineering, Central South University, Changsha 410083, PR China

ARTICLE INFO

Article history:
Available online 31 January 2015

Keywords:
Formal Concept Analysis
Concept lattice
Incremental algorithm
Lattice construction

ABSTRACT

Incremental algorithms for constructing concept lattices can update a concept lattice according to new objects added to the formal context. In this paper, we propose an efficient incremental algorithm for concept lattice construction. The algorithm, called FastAddIntent, results as a modification of AddIntent in which we improve two fundamental procedures including searching for canonical generators and fixing the covering relation. We describe the algorithm completely, prove correctness of our improvements, discuss time complexity issues, and present an experimental evaluation of its performance and comparison with AddIntent. Theoretical and empirical analyses show the advantages of our algorithm when applied to large or (and) dense formal contexts.

1. Introduction

Formal Concept Analysis (FCA) was introduced by Rudolf Wille in the early 1980s (Ganter & Wille, 1999; Wille, 1982, 2009). It is a method of analysis of object-attribute relational data and knowledge representation. For the last two decades, FCA has been used extensively in various disciplines such as software engineering (Wermelinger, Yu, & Strohmaier, 2009), linguistics (Priss, 2005), information retrieval (Dau, Ducrou, & Eklund, 2008), ontology engineering (Maio et al., 2012), bioinformatics (Amin, Kassim, & Hefny, 2013) and data mining (Poelmans, Elzinga, Viaene, & Dedene, 2010). An extensive overview of FCA-based methods in different application domains is given by Poelmans, Ignatov, Kuznetsov, and Dedene (2013a).

As the underlying core structure of FCA, concept lattices have solid mathematical foundations and also the ability to visualize the incidence relationship between objects and attributes. However, applying FCA methods to large formal contexts could bring many challenges, because concept lattices can grow exponentially large in the worst case and counting the number of all concepts is an NP-complete problem (Babin & Kuznetsov, 2010; Kuznetsov, 2001; Kuznetsov, 2004). Since an open problem of "handling large contexts" was pointed out at the fourth international Conference on Formal Concept Analysis, designing more efficient algorithms for handling large and complex incidence matrices has become a popular research topic (Poelmans et al., 2013b), and among all types of those lattice construction algorithms, incremental algorithms have a unique advantage. The input formal context may not be fixed in a real-life application of FCA, which means we have to update the present lattice or compute a new lattice from scratch. Obviously, computing the corresponding changes only and updating the current lattice should be a better choice in most scenarios, which can be handled by incremental algorithms such as Godin's (Godin, Missaoui, & Alaoui, 1995), Object Intersections (Carpineto & Romano, 2004) and AddIntent (Van Der Merwe, Obiedkov, & Kourie, 2004). It makes using incremental algorithms a very suitable and reasonable option for maintaining an online lattice in applications of FCA.

In this paper, we introduce a new incremental algorithm for constructing concept lattices. The algorithm we propose is a refinement of AddIntent (Kourie, Obiedkov, Watson, & van der Merwe, 2009; Van Der Merwe et al., 2004) in which we improve two fundamental procedures including fixing the lattice order relation and searching for canonical generators. The improvements make the algorithm perform considerably better than AddIntent when applied to relatively large or (and) dense datasets.

The remainder of the paper is composed as follows. In Section 2, we recall some basic definitions and propositions of FCA. Section 3 gives a brief survey of incremental algorithms, then describes our algorithm and shows the correctness of the proposed improvements. In Section 4, we discuss complexity issues. Section 5 presents an experimental evaluation of the performance of the presented algorithm. Our work is concluded in Section 6.

2. Preliminaries

In this section, we introduce basic FCA notions and conventions. All definitions and propositions are assumed they are written by

* Corresponding author. Tel.: +86 073182539925.
E-mail addresses: ligeng-zou@csu.edu.cn (L. Zou), zpzhang@csu.edu.cn (Z. Zhang), dragone7968@163.com (J. Long).

http://dx.doi.org/10.1016/j.eswa.2015.01.044
0957-4174/

Ganter and Wille (1999) which the reader is kindly referred to for a more detailed description.

Definition 1. A *formal context* is a triple of sets $K = (G, M, I)$, where $I \subseteq G \times M$ is a binary relation between G and M. The elements in G and M are called *objects* and *attributes*, respectively. gIm or $(g, m) \in I$ indicates the object g has the attribute m.

A formal context can be represented by a cross table (or matrix) where every row is an object and every column is an attribute. Crosses in the table represent the incidence relation I. An example of a simple formal context is illustrated in Table 1.

Definition 2. For a set of objects $A \subseteq G$ we define the set of common attributes shared by all objects in A as:

$$A^{\uparrow I} = \{m \in M | \forall g \in A, gIm\}.$$

Similarly, for a set of attributes $B \subseteq M$ we define the set of objects that have all attributes in B as:

$$B^{\downarrow I} = \{g \in G | \forall m \in B, gIm\}.$$

Definition 3. A *formal concept* of a formal context $K = (G, M, I)$ is defined as a pair (A, B) where $A \subseteq G$, $B \subseteq M$, $A^{\uparrow I} = B$ and $B^{\downarrow I} = A$. A and B are called the *extent* and the *intent* of the concept (A, B), respectively.

Definition 4. Let (A_1, B_1) and (A_2, B_2) be two formal concepts of a given formal context K. (A_1, B_1) is called a *superconcept* of (A_2, B_2) and (A_2, B_2) is called a *subconcept* of (A_1, B_1) if $A_2 \subseteq A_1$ (or equivalently, $B_1 \subseteq B_2$) which can be denoted by $(A_2, B_2) \leqslant (A_1, B_1)$. The set of all formal concepts of K together with the superconcept-subconcept relation makes a complete lattice that is called the *concept lattice* of the context.

Since the subconcept-superconcept relation is a natural partial order relation, we can simply adopt the definition of neighboring nodes of order theory here.

Definition 5. Let c_1 and c_2 be two concepts of a given formal context K. We say c_1 is a *lower neighbor* (or a *child*) of c_2 and c_2 is an *upper neighbor* (or a *parent*) of c_1, if $c_1 \leqslant c_2$ and there is no other concept c_3 with $c_3 \neq c_1$, $c_3 \neq c_2$ and $c_1 \leqslant c_3 \leqslant c_2$. This relationship (also called the *covering relation*) is denoted by $c_1 \prec c_2$.

Like any other partially ordered sets, concept lattices can be represented by line diagrams (or Hasse diagrams). In a line diagram, only neighboring nodes are connected by edges and c_2 should be above c_1 if $c_1 \prec c_2$. For instance, Fig. 1 is the Hasse diagram of the concept lattice derived from Table 1.

In order to make our proposed improvements clear, we need to provide two elementary propositions here.

Proposition 1. *$K = (G, M, I)$ is a formal context where $A, A_1, A_2 \subseteq G$ are sets of objects and $B, B_1, B_2 \subseteq M$ are sets of attributes. Then,*

(1) $A_1 \subseteq A_2 \Rightarrow A_2^{\uparrow I} \subseteq A_1^{\uparrow I}$,
(2) $B_1 \subseteq B_2 \Rightarrow B_2^{\downarrow I} \subseteq B_1^{\downarrow I}$,

Table 1
Example of a formal context.

	a	b	c	d	e
1	×	×	×	×	×
2	×	×	×		×
3			×	×	
4					×
5	×	×	×	×	

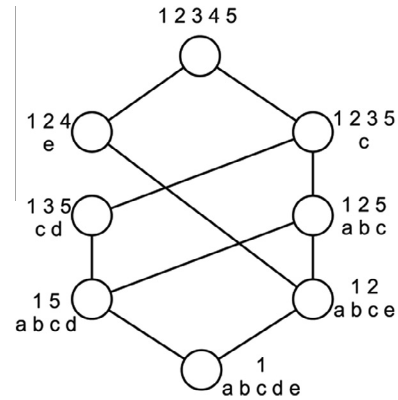

Fig. 1. Concept lattice of the formal context in Table 1.

(3) $A \subseteq A^{\uparrow I \downarrow I}$,
(4) $B \subseteq B^{\downarrow I \uparrow I}$,
(5) $A^{\uparrow I} = A^{\uparrow I \downarrow I \uparrow I}$,
(6) $B^{\downarrow I} = B^{\downarrow I \uparrow I \downarrow I}$,
(7) $A \subseteq B^{\downarrow I} \Leftrightarrow B \subseteq A^{\uparrow I} \Leftrightarrow A \times B \subseteq I$.

Corollary 1. *$A^{\uparrow I \downarrow I}$ is the smallest extent that includes A, and $B^{\downarrow I \uparrow I}$ is the smallest intent that includes B.*

Corollary 2. *$A \subseteq G$ is the extent of a formal concept if and only if $A = A^{\uparrow I \downarrow I}$. Similarly, $B \subseteq M$ is the intent of a formal concept if and only if $B = B^{\downarrow I \uparrow I}$.*

Proposition 2. *T is an index set. For every $t \in T$, let A_t be a set of objects. Then*

$$\left(\bigcup_{t \in T} A_t\right)^{\uparrow I} = \bigcap_{t \in T} A_t^{\uparrow I}.$$

The same statement holds for the sets of attributes too.

3. The FastAddIntent algorithm

3.1. Incremental algorithms

Computing a concept lattice has been widely studied, which leads to the development of many efficient algorithms. In general, we can divide these algorithms into two categories including batch algorithms (Ganter, 2010; Kuznetsov, 1993; Outrata & Vychodil, 2012) and incremental algorithms (Godin et al., 1995; Kourie et al., 2009; Valtchev & Missaoui, 2001; Van Der Merwe et al., 2004). Batch algorithms usually construct the lattice in a bottom-up (or top-down) approach, while incremental algorithms compute the lattice by adding objects (or attributes) of a given context one by one. A significant characteristic of the latter is that any information regarding objects (or attributes) that have not been processed remains unknown to algorithms.

Comparing to batch algorithms, incremental algorithms are more appropriate in real-life applications that work with dynamic datasets and therefore they are of our main interest in this paper. Two of the most representative incremental algorithms are Godin (Godin et al., 1995) and AddIntent (Van Der Merwe et al., 2004). Valtchev, Missaoui, and Godin (2008) adopt a fast variant of the Godin algorithm and integrate it into a very efficient framework for incremental frequent closed itemsets (FCIs) mining. An