Expert Systems With Applications 46 (2016) 224–235

Contents lists available at ScienceDirect

Expert Systems With Applications

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

An efficient algorithm for increasing the granularity levels of attributes in formal concept analysis

Ligeng Zou, Zuping Zhang*, Jun Long

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

ARTICLE INFO

Keywords:
Formal concept analysis
Concept lattice
Granularity of attributes
Interactive data exploration

ABSTRACT

In the basic setting of formal concept analysis, a many-valued attribute needs to be replaced with several one-valued attributes. These one-valued attributes can be interpreted as a certain level of granularity of the corresponding many-valued attribute. In this paper, we explore theoretical relationships between concepts before and after increasing the granularity level of one attribute, based on which we introduce an efficient method of concept classification. Moreover, a new preprocessing routine is proposed to help generate new concepts and restore lattice order relation. These two procedures can considerably reduce the comparisons between sets, compared to the original Zoom-In algorithm. By employing these two procedures, we introduce an efficient algorithm, referred to as Unfold, to increase the granularity levels of attributes. The algorithm can perform a Zoom-In operation on a concept lattice associated with a coarser data granularity to obtain a new one that consists of finer formal concepts without building the new lattice from scratch. We describe the algorithm and present an experimental evaluation of its performance and comparison with another Zoom-In algorithm. Empirical analyses demonstrate that our algorithm is superior when applied to various types of datasets.

1. Introduction

Formal concept analysis (FCA) (Ganter & Wille, 1999; Wille, 2009) is a method for analysis of object-attribute data and knowledge discovery. FCA proved useful in a broad range of applications such as ontology engineering (De Maio, Fenza, Loia, & Senatore, 2012; Kang, Li, & Wang, 2012), information retrieval (Codocedo, Lykourentzou, Astudillo, & Napoli, 2013), software engineering (Li, Sun, & Leung, 2012; Tonella, 2003), mining gene expression data (Amin, Kassim, & Hefny, 2013), linguistics (Priss & Old, 2010) and data mining (La, Le, & Vo, 2014). Comprehensive references to applications of FCA can be found in Poelmans, Ignatov, Kuznetsov, and Dedene (2013).

In the classic setting of FCA, the input data is assumed to be a Boolean matrix where every row is an object and every column is an attribute. However, a binary table containing only 0 s and 1 s is very rare among real datasets, and general attributes in a real dataset could be categorical, ordinal, nominal, etc. In order to transform a data table with general attributes into a one-valued context, FCA uses so-called conceptual scaling to replace a many-valued attribute with several one-valued attributes (Ganter & Wille, 1999). The choice of the scale attributes is usually made by a domain expert according to what values the general attribute may have, which means it is a matter of interpretation. The selected one-valued attributes can be seen as a representation of a certain granularity level of the corresponding attribute, and appropriate granularity levels may reveal particular interesting patterns in the concept lattice (Belohlavek, De Baets, & Konecny, 2014). Moreover, if the concept lattice corresponding to the current granularity level does not expose valuable patterns, one may want to increase or decrease the level of granularity to seek desirable information. Therefore, the capability to change the granularity level of an attribute interactively is necessary and fundamental.

In this paper, we focus on the problem of how to update a concept lattice using a finer granularity level of one attribute. We explore theoretical relationships between concepts before and after increasing the granularity level of the selected attribute. Particularly, we provide sufficient and necessary conditions for identifying different types of concepts. Theoretical bases for fixing the covering relation are also adapted from Van Der Merwe, Obiedkov, and Kourie (2004) and Carpineto and Romano (2004). Based on these theoretical foundations, we propose an efficient algorithm, called Unfold, for increasing the granularity levels of attributes, which includes a unified procedure to determine the type of a concept and a special preprocessing procedure to facilitate the formation of new concepts and the restoration of lattice order relation. Experimental results show that our proposal performs considerably better than the other method in most situations.

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

http://dx.doi.org/10.1016/j.eswa.2015.10.026
0957-4174/

The paper is composed as follows. In Section 2, we recall some basic notions from FCA. Section 3 gives a brief survey of related work on granularity of attributes. Section 4 describes our algorithm and provides theoretical foundations. Section 5 presents an experimental evaluation of the performance of the presented algorithm. Our work is concluded in Section 6.

2. Basic notions from formal concept analysis

In this section, we recall some basic FCA notions and conventions concisely. The reader is kindly referred to Ganter and Wille (1999) for a comprehensive background.

Definition 1. A *formal context* is a triplet $K = (G, M, I)$, 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.

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

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

Correspondingly, for a set $B \subseteq M$ of attributes 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 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 (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 $(A_2, B_2) \leq (A_1, B_1)$. 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 $\leq$ is a partial order, we can 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 $c_1 \leq c_2$ and there is no other concept c_3 with $c_3 \neq c_1$, $c_3 \neq c_2$ and $c_1 \leq c_3 \leq c_2$. This relationship (also called the *covering relation*) is denoted by $c_1 \prec c_2$

Proposition 1. *If $K = (G, M, I)$ is a formal context, $A, A_1, A_2 \subseteq G$ 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) $B_1 \subseteq B_2 \Rightarrow {B_2}^{\downarrow I} \subseteq {B_1}^{\downarrow I}$,
(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. *If T is an index set and, for every $t \in T$, $A_t \subseteq G$ is a set of objects, then*

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

The same holds for sets of attributes too.

3. Related work

3.1. Granularity of attributes

As we discussed in Section 1, a many-valued table has to be transformed into a Boolean matrix using conceptual scaling in the basic setting of FCA. Scaling is a process of replacing each general attribute (e.g. categorical, ordinal, nominal, etc.) with a number of one-valued attributes. For a many-valued attribute, the choice of its corresponding scale attributes in a conceptual scale determines how many details we use to describe the attribute. The selection of appropriate one-valued attributes for a general attribute naturally requires a user to experiment with data interactively for multiple times. If the resulting formal concepts are too specific and expose too many details, the user can choose a coarser granularity level of the attribute. Similarly, one can choose a finer level of granularity to obtain concepts that are more specific.

If a new concept lattice is constructed every time when the user changes the granularity of a general attribute, the process of selecting proper one-valued attributes will be very computationally expensive and time consuming. To the best of our knowledge, this problem was first addressed by Hashemi, De Agostino, Westgeest, and Talburt (2004), and they introduced an algorithm for creating a concept lattice for the coarser data granularity by updating the lattice generated for the finer data granularity. Belohlavek and Sklenar (2005) gives a formal definition of granularity of attributes in FCA. Algorithms that perform Zoom-In/Zoom-Out operations on an original concept lattice can be found in Belohlavek, De Baets, and Konecny (2012). Then, extended versions of algorithms in Belohlavek et al. (2012) are provided by Belohlavek et al. (2014), which exploit disjointness character of a granularity tree. A granularity tree is a hierarchy for conveniently employing granularity of attributes in FCA, which can be formally defined as follows (Belohlavek et al., 2014).

Definition 6. Let X be a set of objects. A *g-tree* (*granularity tree*) for attribute y is a rooted tree with the following properties:

(1) each node of the tree is labeled by a (unique) attribute name: the root is labeled by y;
(2) to each label z of a node a set $z^{\Downarrow} \subseteq X$ is associated; the objects to which attribute z applies;
(3) if the nodes labeled by $z_1, \ldots, z_n$ are the successors of the node labeled by z, then $\{z_1^{\Downarrow}, \ldots, z_n^{\Downarrow}\}$ is a partition of $z^{\Downarrow}$.

A particular level of granularity of attribute y can be described by a cut in a g-tree, which can be defined as follows (Belohlavek et al., 2014).

Definition 7. A *cut* in a g-tree for y is a set S of node labels of the g-tree such that for each leaf node u, there exists exactly one node v on the path from the root y to u such that the label of v belongs to S.

For two cuts $S_1 = \{y_1, \ldots, y_m\}$ and $S_2 = \{z_1, \ldots, z_n\}$ of a given g-tree, we say $S_1 \leq S_2$ if $\{y_1^{\Downarrow}, \ldots, y_m^{\Downarrow}\}$ is a subpartition of $\{z_1^{\Downarrow}, \ldots, z_n^{\Downarrow}\}$, i.e. for every y_i there exits z_j such that $y_i^{\Downarrow} \subseteq z_j^{\Downarrow}$. Hence, S_1 may be seen as a refinement of S_2.

Example 1. For Table 1, let us assume that attribute c is many-valued, while a, b and d are one-valued attributes. Values of c are not given explicitly, for the sake of simplicity. One may consider a simple g-tree for attribute c with a root labeled by c, which is illustrated in Fig. 1. The corresponding sets of objects are given by $c^{\Downarrow} = \{1, 2, 3, 5, 6, 7\}$, $c_1^{\Downarrow} = \{1, 2, 5\}$, $c_2^{\Downarrow} = \{3, 6, 7\}$, $c_3^{\Downarrow} = \{6\}$ and $c_4^{\Downarrow} = \{3, 7\}$. There are three cuts in the g-tree, i.e., $\{c\}$, $\{c_1, c_2\}$ and $\{c_1, c_3, c_4\}$. It is easy to see that $\{c_1, c_3, c_4\} \leq \{c_1, c_2\} \leq \{c\}$. □

Suppose that (G, M, I) is a many-valued context, and we have a g-tree T_y for each attribute $y \in M$. Let S_y be a cut in T_y for each $y \in M$.