Knowledge Discovery from Very Large Databases Using Frequent Concept Lattices

Kitsana Waiyamai[1] and Lotfi Lakhal[2]

[1] Faculty of Engineering, Kasetsart University, Bangkok Thailand
[2] LIMOS, University of Clermont-Ferrand II, France

Abstract. Data clustering and association rules discovery are two related problems in data mining. In this paper, we propose to integrate these two techniques using the *frequent concept lattice* data structure - a formal conceptual model that can be used to identify similarities among a set of objects based on their frequent attributes (frequent items). Experimental results show that clusterings and association rules are generated efficiently from the frequent concept lattice, since response time after lattice construction is measured almost in seconds.

1 Introduction and Motivation

Recently, several clustering methods have been developed in the framework of the concept lattice [1,4,2] which focus on discovering all possible concepts. These methods are inefficient under the context of large databases. First, they have been designed to work in main memory with small datasets, thus limiting their suitability for data mining in large databases. Second, most of them perform an exhaustive search of all possible concepts, whereas only part of them is considered useful by users [5]. An efficient learning method in a real-world context, that does not carry out an exhaustive search of the whole concept lattice, must be provided. The idea of generating associations between items from concept lattices has also been addressed early in several works, e.g., [11,3]. However, the rules generated by these methods are particular cases of association rules, i.e. they are association rules with confidence equal to 100%. Further, their associated algorithms work only in main memory.

In this paper, we propose a data mining framework using a concept lattice [10] as a tool for the knowledge discovery, with an emphasis on the integration of data clustering and association rule discovery from large databases. The idea is to preprocess the database and derive a *frequent concept lattice* - a data structure that encodes the information needed for our discovering tasks. A frequent concept lattice is a part of the concept lattice in which each concept covers at least some initial minimum number of objects of the database, that say the items (in the concept's intent) shared by those objects in the concept's extent must have a support greater or equal the user-specified initial support *init_sup*. The support of an itemset is defined as the percentage of objects in the database containing that itemset.

R. López de Mántaras, E. Plaza (Eds.): ECML 2000, LNAI 1810, pp. 437–445, 2000.

From the frequent concept lattice, *pertinent concepts* and *strong association rules*, with respect to the user point of view, are generated. We developed a collection of operators to generate all pertinent concepts and strong association rules directly from the frequent concept lattice without further access to the original database. Experimental results show that knowledge are discovered from the frequent concept lattice, with response time after lattice construction measured almost in seconds. The rest of the paper is organized as follows: Section 2 begins by formally defining the frequent concept lattice model. Then, an algorithm for building a frequent concept lattice from a given a database is described. Section 3 provides experimental results with data extracted from the statistical Census data of Kansas USA[1]. Section 4 concludes with a summary and future works.

2 Frequent Concept Lattice Based Data Clustering

In this section, we begin by formally defining the frequent concept lattice model. Then, we discuss a process for identifying clusters form a data mining context based on the frequent concept lattice construction.

2.1 The Frequent Concept Lattice Model

Following, we give the formal definitions of *data mining context*, *Galois connection*, *concept*, *frequent concept* and finally *frequent concept lattice*.

Data mining context A data mining context (a database) is defined as a triple $\mathcal{D} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$, where $\mathcal{O}$ and $\mathcal{I}$ are finite sets of objects (or transactions) and database items, $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{I}$ is a binary relation. Each couple (o, i) of $\mathcal{R}$ denotes the fact that the object o of $\mathcal{O}$ has the item i of $\mathcal{I}$. Figure 1 and 2 represent the data mining context example using horizontal and vertical representations.

Object ID	*Items*
1	{A F}
2	{A F G}
3	{A B F G}
4	{B F G H}
5	{A C E}

Fig. 1. Data mining context using a horizontal representation

Item	*OIDs*
A	**{1 2 3 5}**
B	**{3 4}**
C	{5}
D	{}
E	{5}
F	**{1 2 3 4}**
G	**{2 3 4}**
H	{4}

Fig. 2. Data mining context using a vertical representation

Galois connection Let $\mathcal{D} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$ be a data mining context. For $O \subseteq \mathcal{O}$ and $I \subseteq \mathcal{I}$, we define $f : 2^{\mathcal{O}} \rightarrow 2^{\mathcal{I}}, f(O) = \{i \in \mathcal{I} \mid \forall \mathrm{o} \in O \cdot \mathrm{o}\mathcal{R}\mathrm{i}\}$ and conversely $g : 2^{\mathcal{I}} \rightarrow 2^{\mathcal{O}}, g(I) = \{\mathrm{o} \in \mathcal{O} \mid \forall \mathrm{i} \in I \cdot \mathrm{o}\mathcal{R}\mathrm{i}\}$. That is, $f(O)$ is the set of all items, called *itemset*, common to all objects in O, and $g(I)$ is the set of

[1] `ftp://ftp2.cc.ukans.edu/pub/ippr/census/pums/pums90ks.zip`

all objects which have all items in I. The pair (f, g) forms a Galois connection between $2^{\mathcal{O}}$ and $2^{\mathcal{I}}$, the power sets of $\mathcal{O}$ and $\mathcal{I}$, respectively. Furthermore, the compositions $f \circ g$ and $g \circ f$[2] are closure operators.

Concept A concept of the data mining context $\mathcal{D} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$ is defined as a pair $(Extent, Intent)$, where $Extent \subseteq \mathcal{O}$, $Intent \subseteq \mathcal{I}$, $f(Extent) = Intent$ and $g(Intent) = Extent$. Hence, a concept c is formed by two parts: an extent represents a subset of objects, denoted as $Extent(c)$, and an intent represents the common items between this subset of objects, denoted as $Intent(c)$. That is, a concept is a maximal collection of objects sharing common items.

Frequent concept Let L be a set of all concepts formed from $\mathcal{D} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$, and c a concept of L. Let *init_sup* be a user-specified initial support. A *support* threshold associated to a concept c, denoted as $supp(c)$, is the percentage of objects in $\mathcal{O}$ having precisely the same set of items in the intent of c, which we can be defined as: $supp(c) = \frac{\|Extent(c)\|}{\|\mathcal{O}\|}$ A concept is said *frequent* if its support threshold is greater or equal to *init_sup*.

Frequent concept lattice Let L be a set of all concepts formed from $\mathcal{D} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$, *init_sup* a user-specified initial support, and FL a set of all frequent concepts, i.e. $FL = \{c \in L \mid supp(c) \geq init\_sup\}$. The frequent concept lattice $\mathcal{FL} = (FL \cup \{\bot\}, \leq)$ of a data mining context $\mathcal{D}$, is a complete lattice of frequent concepts derived from $\mathcal{D}$. *Proof* of this property is given in [8]. Figure 3 shows the frequent concept lattice (using a graph-oriented representation) derived from the data mining context example with *init_sup* equal to 35% (at least 2 objects are contained in each concept).

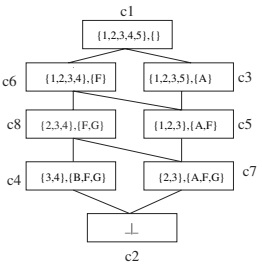

Fig. 3. Graph-oriented representation of the *frequent concept lattice* with the support threshold $\geq$ 35%(2/5)

Concept-ID	Extent	Intent
c1	{1,2,3,4,5}	{}
c2	{}	{A,B,F,G}
c3	{1,2,3,5}	{A}
c4	{3,4}	{B,F,G}
c5	{1,2,3}	{A,F}
c6	{1,2,3,4}	{F}
c7	{2,3}	{A,F,G}

Fig. 4. Database-oriented representation (nested relation) of the *frequent concept lattice*

2.2 The Lattice Clustering Algorithm

In contrast with existing works [11,4,6,5,2] where concept lattices are represented using a graph-oriented representation (where sub-superconcept relationships between concepts are explicitly represented), we address concept lattice representation in database systems from a set-oriented perspective. In our approach,

[2] We use the following notations: $f \circ g(O) = g(f(O))$ and $g \circ f(I) = f(g(I))$.

concept lattices are represented as finite sets of concepts in which *concept lattice constraints* [9] are given, insuring that the sets effectively preserve the lattice structures. Figure 4 shows the frequent concept lattice using a database-oriented representation. From a data mining context, we want to derive the frequent concept lattice associated to that context, that is to say finding all the frequent concepts (Ext, Int) such that Ext is the maximal set of objects which all have precisely the same set of items in Int and such that the number of those objects represents a percentage no less than the initial support *init_sup*. The idea is to successively add a new frequent item (i.e. item that has its support greater or equal to the initial support *init_sup*) to the current frequent concept lattice. Formal characterization of inserting a new frequent item into a frequent concept lattice is given in [8].

procedure Insert_frequent_item(i: new frequent item, $g(\{i\})$: object set containing i, *init_sup*: initial support);
begin
0) $Exist = \{\}$; // $Exist$ contains extended concepts or new concepts
1) **for** $j = |Intent(\bot)|$ **downto** $|Intent(Sup)|$ **do**
2) $levelj$ = **select** c **from** c **in** $\mathcal{FL}$ **where** $|Intent(c)| == j$;
3) **for all** concept $c \in levelj$ **do**
4) **if** $(Extent(c) \subseteq g(\{i\}))$ **then** // c is a extended frequent concept
5) $Intent(c) = Intent(c) \cup \{i\}$;
6) **if** $(Extent(c) == g(\{i\}))$ **then exit** procedure **endif**;
7) $Exist = Exist \cup Extent(c)$;
8) **else**
9) $inter = g(\{i\}) \cap Extent(c)$;
10) **if** $(|inter| \geq (init\_sup \times |\mathcal{D}|)$ and $inter \notin Exist)$ **then**
11) Create a new frequent concept nc;
12) $Extent(nc) = inter$; $Intent(nc) = Intent(c) \cup \{i\}$;
13) $\mathcal{FL} = \mathcal{FL} \cup \{\text{nc}\}$;
14) **if** $(inter == g(\{i\}))$ **then exit** procedure **endif**;
15) $Exist = Exist \cup inter$;
16) **endif**;
17) **endif**;
18) **end**;
19) **end**;
end

Algorithm Lattice_clustering(*init_sup*: initial support, $\mathcal{D}$: data mining context);
begin
0) Initialize Sup and $\bot$;
1) $\mathcal{FL} = \{\text{Sup}, \bot\}$;
2) **for** $k = 1$ **to** $|\mathcal{I}|$ **do**
3) **if**$(supp(i_k) \geq init\_sup)$ **then**
4) $Insert\_frequent\_item(i_k, g(\{i_k\}), init\_sup)$;
5) **endif**;
6) **end**;
end

The algorithm for building a frequent concept lattice from a database given above takes as input (i) an item i to be inserted, (ii) its associated object set $g(\{i\})$, (iii) initial support *init_sup*, (iv) a frequent concept lattice $\mathcal{FL}$ to be updated, and a database $\mathcal{D}$. It starts by initializing the lattice with the Top and $\perp$ elements. The Top element corresponds to the most general (with respect to set inclusion) frequent concept in the lattice, i.e. its extent contains all the objects of the database All). The $\perp$ element corresponds to the most specific (with respect to set inclusion) frequent concept in the lattice, i.e. its intent contains all the frequent items of the database All). Then, the construction of $\mathcal{FL}$ consists of successively calls to the procedure $Insert\_frequent\_item$ which inserts a new frequent item i to the current frequent concept lattice. Each insertion may discover new frequent concepts and/or augment the intents of existing frequent concepts. The main loop of the procedure *Insert_frequent_item* (lines 1-19) visits the concepts of the lattice $\mathcal{FL}$ level by level in decreasing cardinality of their intent. At each level, an SQL query is executed on $\mathcal{FL}$ in order to load all the concepts with the same cardinality of their intent in main memory (line 2 of the procedure). Then, it examines successively all the concepts of the same level (lines 3-18 of the procedure). For each concept c of $\mathcal{FL}$, the procedure tests how it relates to $g(\{i\})$:

- if $Extent(c) \subseteq g(\{i\})$ (i.e. its extent is more general than or equal to the set of objects associated to the new item i), c is an *augmented concept*, then its intent is augmented by the new item i (lines 4-5 of the procedure),
- else the new extent $inter = Extent(c) \cap g(\{i\})$ is calculated. To verify if $inter$ is not already appeared in any concept of $\mathcal{FL}$, just examine all the frequent concepts that have been augmented or created previously: that's the role of the list $Exist$ which keeps all concepts newly augmented or created. If $inter \notin Exist$ and $|inter| \geq (init\_sup \times |\mathcal{O}|)$, then c is a generator of the *new frequent concept* $(inter, Intent(c) \cup \{i\})$. This is valid because all existing concepts are treated in decreasing cardinality of their intent, the first concept encountered which gives a new intersection is the generator of the new concept because it is necessary the smallest concept (with respect to number of its objects).

The execution of the procedure *Insert_frequent_item* terminates when a frequent concept is encountered (line 6) or created (line 14) for which its extent is equal to $g(\{i_k\})$. Using the database example (c.f. Figure 2), the iterative steps of building a frequent concept lattice with initial support = 35%(2/5), i.e. each concept of the lattice must covers at least 2 objects of the database, is illustrated[3] below in Figure 5. The algorithm starts by initializing the lattice with the two elements: $Sup = c1 = (All, \emptyset)$ and $\perp = c2 = (\{3\}, All)$ (line 0). Then, for each frequent item, the procedure *Insert_frequent_item* is called. Inserting the item A results in the insertion of a new frequent concept $c3$ generated by the concept generator $c1$. Inserting the item B results in discovering of two new frequent concepts:

[3] We use a Graph-oriented representation of the lattice in order to facilitate understanding.

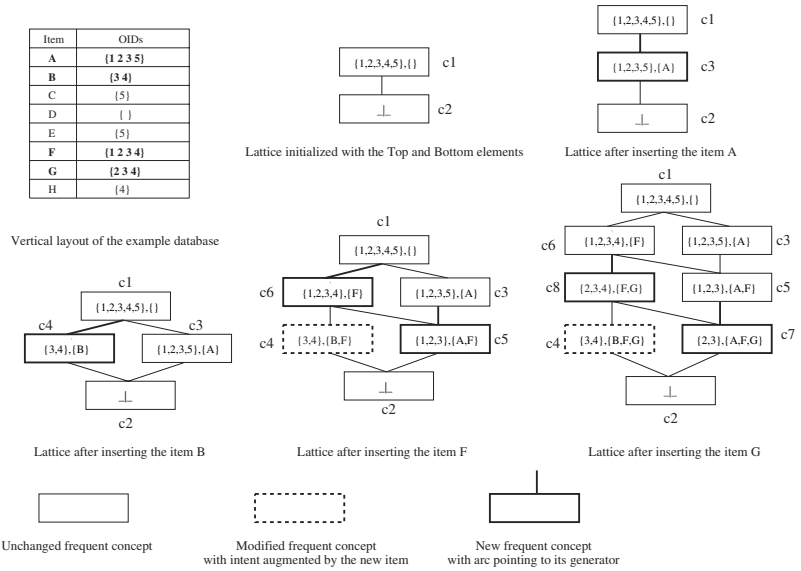

Fig. 5. Iterative steps of building a frequent concept lattice with initial support = 35%(2/5)

$(\{3\}, \{A, B\})$ generated by $(\{1, 2, 3, 5\}, \{A\})$ and $(\{3, 4\}, \{B\})$ generated by $c1$. However only $c4 = (\{3, 4\}, \{B\})$ is added to the lattice since its support threshold is greater than the initial support *init_sup* given by the users. Inserting the item F results in the insertion of two new frequent concepts: $c5$ generated by $c3$ and $c6$ generated by $c1$, and the modification of the old frequent concept: $c4$. Inserting the item G results in the insertion of two new frequent concepts: $c7$ generated by $c5$ et $c8$ generated by $c6$, and the modification the old frequent concept: $c4$.

3 Knowledge Discovering from the Frequent Concept Lattice

Once constructed, the frequent concept lattice can be used as a support for generating pertinent concepts and strong association rules. We propose a collection of operators for the tasks of knowledge discovering: lattice operators (UB,LB,MEET,JOIN, ...) and rule discovery operator. For further details of the formal definitions of all these operators, interested reader should consult ([9,8,7]). To assess their relative performances of the proposed operators, our algorithms were implemented in O_2C object database programming language provided by the O_2 OODBMS. The platform we used was a 43P240 bi-processor IBM PowerPC running AIX 4.1.5 and O_2 system version 4.6 with a CPU clock rate of 166 MHz, 1GB of main memory and a 9 GB disk. Only one processor was used since

the application was single-threaded. The test program was allowed a maximum of 128 MB. Swapping and buffering mechanisms are provided by the O_2 system. We ran our tests using the Census data which belong to the domain of statistical databases. Our objective is to classify the set of persons according to several characteristics such as sex, age, profession, etc. The Census data were extracted from the Kansas 1990 PUMS file (Public Use Microdata Samples). We took two datasets containing the first 100000 persons: c10d100k and c15d100k. Each person in c10d100k has 10 items (from the total of 281 items). Each person in 15d100k has 15 items (from the total of 309 items). We treat each person as a single transaction, where the items are the characteristics associated to persons.

3.1 Frequent Concept Lattice Construction

Although in the worst case, the size of a frequent concept lattice (i.e. the number of concepts it contains) can be exponential with respect to the number of database objects, its size is linear with respect to the number of database objects when there exists an upper bound k which is the average size (number of items) of a database object. In our experiments, k was set to respectively 10 and 15. Figure 6 shows the CPU time (including disk access) measured the total time (in second) necessary to build the lattice by adding new items one by one calling the lattice clustering algorithm (c.f. Figure 2.2). The important difference between the two curves can be explained by the difference in the size of the two resulting lattices. Indeed, the algorithm visits almost all concepts of the lattice and performs the intersection operation at each encountered concept. A more judicious implementation, without visiting every concept of the lattice, should accelerate the construction time.

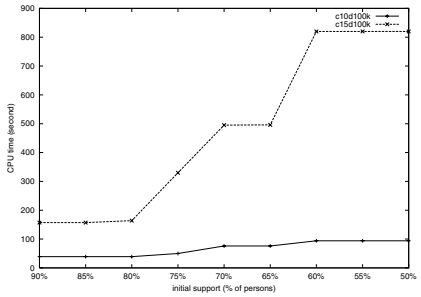

Fig. 6. CPU time of frequent concept lattice construction on c10d100k and c15d100k with different supports

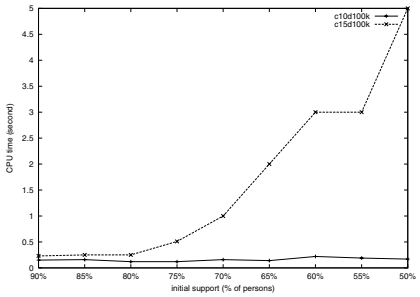

Fig. 7. CPU time of discovering association rules from the two lattices with different supports

3.2 Querying Strong Association Rules

At this step, the frequent concept lattice is constructed and stored in the O_2 database system, we can then use it to generate all frequent itemsets and then derive all association rules. Experiments were conducted on the two databases c10d100k and c15d100k using different minimum supports ranging form 50% to 90% to get meaningful response times. The minimum confidence is fixed to 75%. In Figure 7, one can see the running times of the experiments. Logically, we observe that the bigger the support is the shorter the association rule generation time.

4 Conclusion and Future Works

This paper proposes a framework to integrate data clustering and association rule discovery. The heart of the framework is the use of frequent concept lattice data structure during the process of knowledge discovery. Experimental results show that our method can generate pertinent concepts and strong association rules efficiently, since response time after lattice construction is measured almost in seconds. In our future work, we will focus on updating techniques for maintenance of the frequent concept lattice to handle dynamic databases. This approach avoids the repetitive process of discovering all frequent concepts from crash each time a new object is introduced in the databases, and allows incremental data clustering and association rule discovery.

References

1. I. Bournaud and J.G. Ganascia. Conceptual clustering of complex objects: A generalization space based approach. In 3^{th} *Inter. Conf. on Conceptual Structures (ICCS'95)*, 1995. 437
2. C. Carpineto and G. Romano. A lattice conceptual clustering and its application to browsing retrieval. *Machine Learning*, (24):95–122, 1996. 437, 439
3. R. Godin and R. Missaoui. An incremental concept formation approach for learning from databases. *Theoretical Computer Science: Special Issue on Formal Methods in Databases and Software Engineering*, 133:387–419, 1994. 437
4. R. Godin, R. Missaoui, and H. Alaoui. Incremental Concept Formation Algorithms based on Galois (Concept) Lattices. *Computational Intelligence*, 11(2):246–267, 1995. 437, 439
5. T.B. Ho. An approach to concept formation based on formal concept analysis. *IEICE Trans. Information and Systems*, (5):553–559, 1995. 437, 439
6. R.E. Kent. Automatic classification. Technical report, Department of Computer and Information Science, University of Arkansas, 1995. 439
7. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Journal of Information Systems*, 24(1):25–46, 1999. 442
8. K. Waiyamai. *Knowledge Discovery and Management from very large Databases: A Frequent Concept Lattice Approach (In French)*. Ph.d. thesis, University of Clermont-Ferrand II, France, 1999. 439, 440, 442

9. K. Waiyamai, R. Taouil, and L. Lakhal. Querying concept lattices in object databases. In 3^{th} *Biennial World Conference on Integrated Design & Process Technology (IDPT'98)*, pages 280–289, Berlin, Germany, 1998. 440, 442
10. R. Wille. Restructuring lattices theory: an approach based on hierarchies of concepts. In I. Rival, editor, *Ordered Sets*, pages 445–470. Dordrecht-Boston, 1982. 437
11. R. Wille. Concept lattices and conceptual knowledge systems. In *Computers Math. Applications*, volume 23, pages 493–515. 1992. 437, 439