

CoMine: Efficient Mining of Correlated Patterns *

Young-Koo Lee, Won-Young Kim, Y. Dora Cai, and Jiawei Han

University of Illinois at Urbana-Champaign, Illinois, U.S.A.

{yklee, wykim, doracai, hanj}@uiuc.edu

Abstract

Association rule mining often generates a huge number of rules, but a majority of them either are redundant or do not reflect the true correlation relationship among data objects. In this paper, we re-examine this problem and show that two interesting measures, *all_confidence* (denoted as α) and *coherence* (denoted as γ), both disclose genuine correlation relationships and can be computed efficiently. Moreover, we propose two interesting algorithms, *CoMine*(α) and *CoMine*(γ), based on extensions of a pattern-growth methodology. Our performance study shows that the *CoMine* algorithms have high performance in comparison with their Apriori-based counterpart algorithms.

1. Introduction

Association rule mining is a widely studied topic in data mining research. Its popularity can be attributed to the simplicity of the problem statement and the efficiency of its initial algorithms, such as Apriori [1]. Unfortunately, association mining with real data sets is not so simple. If the *min_support* threshold is high, only commonsense “knowledge” will be found. However, when *min_support* is set low, a huge number of association rules will usually be generated, a majority of which are redundant or noninformative. The true correlation relationships among data objects are buried deep among a large pile of useless rules.

To overcome this difficulty, *correlation* has been adopted as an interesting measures since most people are interested in not only association-like co-occurrences but also the possible strong correlations

implied by such co-occurrences. Brin et al. [2] introduced *lift* (called *interest* there) and a χ^2 correlation measure and developed methods for mining such correlations. However, since these measures based on sophisticated statistical computations do not have the *downward closure property* [1], some approximate similarity measures has to be proposed to explore efficient computation. Ma and Hellerstein [5] proposed a mutually dependent patterns and an Apriori-based mining algorithm. Recently, Omiecinski [7] introduced two interesting measures, called *all_confidence* and *bond*. Both have the downward closure property. Experimental results on some real data sets show that these measures are appealing since they reduce significantly the number of patterns mined, and the patterns are coherent.

In this study, we contribute to mining strongly correlated patterns in two aspects. First, several popularly used measures related to correlation mining are re-examined. Our discussion shows that both *all_confidence* (denoted as α) and *coherence* (called *bond* in [7]) (denoted as γ here) are good measures and in most cases are better than the popularly used *lift* and χ^2 measures. Second, we develop two interesting algorithms, *CoMine*(α) and *CoMine*(γ), by extension of a pattern-growth mining methodology [3]. Our experimental and performance study shows that both *CoMine* algorithms generate truly interesting correlation patterns and have better performance over their Apriori-based counterpart algorithms, *Partition* [7].

2 What measures are good for mining correlated patterns?

In this section, we examine which of these four measures: *all_confidence*, *coherence*, *lift*, and χ^2 , is more suitable at expressing correlations among items in a transactional database. We proceed by first introducing a few concepts.

Let $I = \{i_1, i_2, \dots, i_m\}$ be a set of items, and DB be a database that consists of a set of transactions. Each

*The work was supported in part by National Science Foundation under Grant No. 02-09199, the Univ. of Illinois, and an IBM Faculty Award. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.

transaction T consists of a set of items such that $T \subseteq I$. Each transaction is associated with an identifier, called *TID*. Let A be a set of items, referred to as an *itemset*. An itemset that contains k items is a k -*itemset*. A transaction T is said to *contain* A if and only if $A \subseteq T$. The *support* of an itemset X in DB, denoted as $sup(X)$, is the number of transactions in DB containing X . An itemset X is *frequent* if it occurs no less frequent than a user-defined *min_support* threshold.

Definition 2.1 Given an itemset $X = \{i_1, i_2, \dots, i_k\}$, the universe, the *max_item_sup*, *all_confidence*, and the coherence of X are defined as,

$$universe(X) = \{t_i \in DB | \exists i_j \in X (i_j \in t_i)\} \quad (1)$$

$$max_item_sup(X) = max\{sup(i_j) | \forall i_j \in X\} \quad (2)$$

$$all_conf(X) = \frac{sup(X)}{max_item_sup(X)} \quad (3)$$

$$coh(X) = \frac{sup(X)}{|universe(X)|} \quad (4)$$

Two other measures, *lift* and χ^2 , have been popularly used. χ^2 follows the standard definition in statistics [6], whereas the *lift* of two itemsets A and B is defined as [2],

$$lift(AB) = \frac{sup(AB)}{sup(A)sup(B)} \quad (5)$$

Given a transaction database DB , a *minimum support* threshold *min_support*, a *minimum all_confidence* threshold *min_alpha*, and a *minimum coherence* threshold *min_gamma*, a frequent itemset X is **all_confident** if $all_conf(X) \geq min_alpha$; and it is **coherent** if $coh(X) \geq min_gamma$.

Example 1 The correlation relationships between the purchases of two items, *milk* and *coffee*, can be examined by summarizing their purchase history in the form of Table 1, a 2×2 contingency table, where an entry such as *mc* represents the number of transactions containing both milk and coffee. Table 1 shows the concrete contingency tables and the four measures for a set of transaction database A_1 to A_4 . From the table, one can see that m and c are strongly positively correlated in A_1 but strongly negatively correlated in A_2 . However, λ (*lift*) and χ^2 are very poor indicators, whereas α and γ are very good ones. Similarly, A_3 is negatively correlated, which cannot be shown by λ and χ^2 but clearly shown by α and γ values. Finally, values in A_4 are independent, which is correctly shown in λ , α , and χ^2 and approximated in γ . The reason that λ and χ^2 are so poor in many cases is because they are strongly

influenced by the value of \overline{mc} which is usually big and across a wide range (many people may buy neither m nor c). On the other hand, the definitions of α and γ remove the influence of \overline{mc} , playing a similar role as Jaccard coefficient in cluster analysis [4]. ■

Our experiments using a real database called the CRIME data set¹ show that mining for all_confident or coherent patterns generates a much smaller set but truly correlated patterns in real databases. We omit the details due to lack of space.

3. Mining All-Confident Patterns

Given *min_support* and *min_alpha*, the set of all_confident patterns can be computed in the spirit of the Apriori algorithm. However, as many studies have shown [3], Apriori may not be very efficient when a database is large and/or when patterns to be mined are numerous and/or long because it may generate a huge set of candidate patterns. Therefore, we will examine how to develop more efficient methods for evaluation of such measures. This leads to the development of CoMine algorithms by extension of the pattern-growth methodology, represented by FP-growth [3].

Algorithm CoMine(α) is described briefly as follows. First, we build an FP-tree, in which the labels on each path in FP-tree and the entries in a header table of the tree are ordered in the *f_list* order, the order of descending support counts [3]. Then, we perform tree mining by traversing down the header table. Algorithm 1 shows the tree mining algorithm for the α conditional FP-tree. For each item in the header of the tree, repeat the following. Examine the items in the header in a top-down manner. In Algorithm 1, Line 2 generates the all_confidence pattern β . Since the FP-tree is constructed in such a way that every β becomes an all_confidence pattern, we do not need to check whether β satisfies the conditions. Lines 3 and 4 compute count for each item in β -projected database. Lines 5–8 conduct pruning based on *min_support* and *min_alpha*. Thus, β 's conditional database contains only the items b_j 's such that βb_j satisfies both the minimum support and all_confidence thresholds.

When computing count for each item in β -projected database, CoMine(α) reduces the search space using the properties of the all_confidence measure. In the β -conditional database, for item x to be included in an all_confidence pattern, the support of αx (= the

¹The CRIME data set can be obtained from a crime report database from a Web site related to NIBRS (National Incident-Based Reporting System) (<http://www.icpsr.umich.edu/NACJD/NIBRS>)

	milk	\neg milk	row_sum
coffee	mc	\overline{mc}	c
\neg coffee	$m\overline{c}$	$\overline{m\overline{c}}$	\overline{c}
col_sum	m	\overline{m}	Σ

DB	mc	\overline{mc}	$m\overline{c}$	$\overline{m\overline{c}}$	λ	α	γ	χ^2
A_1	1000	100	100	10,000	9.26	0.91	0.83	9055
A_2	100	1000	1000	100,000	8.44	0.09	0.05	670
A_3	1000	100	10000	100,000	9.18	0.09	0.09	8172
A_4	1000	1000	1000	1000	1	0.5	0.33	0

Table 1. Contingency tables and a few measures.

Algorithm 1 CoMine(α): Mining all-confidence patterns by extending the FP-growth method

Procedure FP-mine($Tree, \alpha$)

- 1: **for** each a_i in the header of $Tree$ **do**
- 2: generate pattern $\beta = \alpha \cup a_i$ with all_conf = $sup(\beta)/max_item_sup(\beta)$; $\{sup(\beta) = sup(a_i)$ in α -projected database}
- 3: get a set I_β of items to be included in β -projected database;
- 4: for each item in I_β , compute its count in β -projected database;
- 5: **for** each b_j in I_β **do**
- 6: if $sup(\beta b_j) < min_support$, delete b_j from I_β ; $\{\text{pruning based on minimum support}\}$
- 7: if $all_conf(\beta b_j) < min_alpha$, delete b_j from I_β ; $\{\text{pruning based on minimum all-confidence}\}$
- 8: **end for**
- 9: construct β -conditional FP-tree with items in I_β $Tree_\beta$;
- 10: **if** $Tree_\beta \neq \emptyset$ **then**
- 11: call FP-mine($Tree_\beta, \beta$)
- 12: **end if**
- 13: **end for**

support of x in the α -conditional database) should be no less than $min_alpha \times max_item_sup(\beta x)$. With this pruning rule, we can reduce the set of items I_β to be counted and, thus, reduce the number of nodes visited when traversing the FP-tree to count each item in I_β .

4. Mining Coherent Patterns

In this section we present an efficient coherent pattern mining algorithm, called CoMine(γ). Unlike the extension of FP-growth for mining all-confidence patterns, the extension for mining coherent patterns has a challenging problem: computing the cardinalities of the universes for given itemsets using the FP-tree.

We present a novel method for computing the cardinality values by exploiting the characteristics of the FP-tree. Let $X = a_1 a_2 \dots a_n$ be an itemset and A_i be the TID set for item a_i . We can compute $|universe(a_1, a_2, \dots, a_n)|$ by computing

Data set	#Tuples	#Items	ATL/MTL
gazelle	59602	497	2.5/267
pumsb	49046	2113	74/74

Table 2. Characteristics of Real Data sets.

$|universe(a_1, a_2, \dots, a_{n-1})| + |A_n| - |(A_1 \cup A_2 \cup \dots \cup A_{n-1}) \cap A_n|$. Here, $|universe(a_1, a_2, \dots, a_{n-1})|$ has been computed in the previous step since we follow the top-down approach. $|A_n|$ can be obtained by maintaining the support for each item. We can compute $|(A_1 \cup A_2 \cup \dots \cup A_{n-1}) \cap A_n|$ by adding the values of the nodes of label a_n that has any of the item a_1, a_2, \dots, a_{n-1} as an ancestor using node-link and parent link of FP-tree[3].

Although we can compute the size of the universe using the FP-tree, the cost is still non-negligible. We can reduce the number of computations by first checking the following condition: $\frac{sup(X)}{|universe(a_1, a_2, \dots, a_{n-1})|} \geq min_gamma$. If this condition fail, there is no need to compute the size of universe.

5. Experiments

In this section, we report our experimental results on the performance of CoMine in comparison with an efficient Apriori-based counterpart algorithm, *Partition* algorithm[7]. Our implementation of *Partition* algorithms also contains optimization techniques proposed in [5, 8]. The result shows that CoMine outperforms *Partition*, and it is efficient and scalable for mining correlated patterns in large databases. Experiments were performed on a 2.2GHz Pentium IV PC with 512MB of memory, running Windows 2000. Algorithms were coded with Visual C++.

Our experiments were performed on real data sets as shown in Table 2. Gazelle comes from click-stream data from Gazelle.com and pumsb is obtained from www.almaden.ibm.com. The gazelle is rather sparse in comparison with pumsb, which is very dense so that it produces many long frequent itemsets even for very high values of support.

Let us compare the relatively efficiency the *Partition*

and CoMine methods on a transactional data set. Figure 1 shows the performance on the gazelle data set when min_sup is fixed at 0.1% and min_conf varies. As we can see in these figures, CoMine always outperforms counterpart Partition for the coherence and all_confidence measures over the entire range of confidence threshold. This is because as the support or confidence threshold goes down, the number as well as the length of frequent itemsets increases and FP-growth-based methods have better scalability than Apriori-based methods under these circumstances.

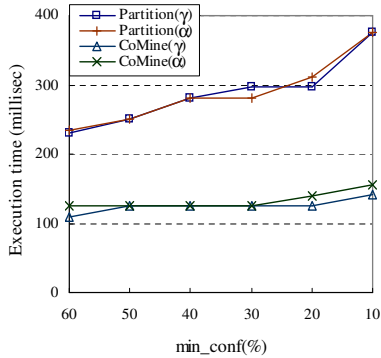


Figure 1. Gazelle when $min_sup = 0.1\%$.

Figure 2 shows the execution time on the pumsb data set using different confidence threshold when the min_sup is fixed at 5%. It also shows that CoMine algorithms always outperform Partition algorithms. When min_conf is less than 75%, Partition algorithms run out of memory and cannot finish. At min_conf 80%, CoMine(α) can be more than 300 times faster than Partition(α) and CoMine(γ) can be more than 1000 times faster than Partition(γ). Figure 2 indicates that the gaps between CoMine(γ) and CoMine(α) are quite small compared with that of Partition(γ) and Partition(α) (Notice that the vertical axis is on a log scale). This is because CoMine(γ) uses the efficient algorithm of computing cardinalities of universes and pruning rules.

6. Conclusions

In this paper, we show that mining all_confidence and coherent patterns generates highly correlated patterns and are better measures than association-based confidence measure and two other claimed correlation measures: lift and χ^2 . For efficient mining of the two kinds of correlation patterns, we proposed a new method, called CoMine, which consists of two algorithms CoMine(α) and CoMine(γ) for computing

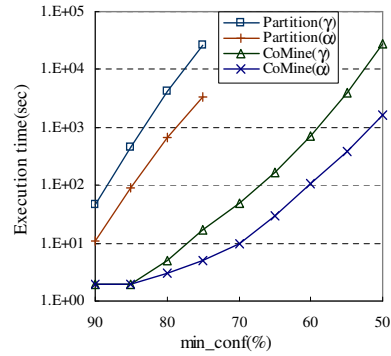


Figure 2. Pumsb when $min_sup = 5\%$.

all_confident and coherent patterns respectively. For CoMine(γ), the proposed method can compute the cardinality of the universe efficiently using the FP-tree structure only. Several pruning methods are also developed that reduce the search space. Our extensive performance study shows that CoMine generates desirable correlation patterns and outperforms the previously proposed Apriori-based algorithms.

References

- [1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. VLDB*, Sept. 1994.
- [2] S. Brin, R. Motwani, and C. Silverstein. Beyond market basket: Generalizing association rules to correlations. In *Proc. SIGMOD*, May 1997.
- [3] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. SIGMOD*, May 2000.
- [4] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley & Sons, 1990.
- [5] S. Ma and J. L. Hellerstein. Mining mutually dependent patterns. In *Proc. ICDM*, Nov. 2001.
- [6] S. Morishita and J. Sese. Traversing itemset lattice with statistical metric pruning. In *Proc. PODS*, May 2001.
- [7] E. Omiecinski. Alternative interest measures for mining associations. *IEEE Trans. Knowledge and Data Engineering*, 15:57–69, 2003.
- [8] H. Xiong, P.-N. Tan, and V. Kumar. Mining hyperclique patterns with confidence pruning. In *Tech. Report*, Univ. of Minnesota, Minneapolis, March 2003.