Flexible Mining of Association Rules

Hong Shen
Japan Advanced Institute of Science and Technology, Japan

INTRODUCTION

The discovery of association rules showing conditions of data co-occurrence has attracted the most attention in data mining. An example of an association rule is the rule "the customer who bought bread and butter also bought milk," expressed by *T(bread; butter)→ T(milk)*.

Let $I=\{x_1,x_2,\ldots,x_m\}$ be a set of (data) items, called the domain; let D be a collection of records (transactions), where each record, T, has a unique identifier and contains a subset of items in I. We define *itemset* to be a set of items drawn from I and denote an itemset containing k items to be k-itemset. The support of itemset X, denoted by $\tilde{A}(X/D)$, is the ratio of the number of records (in D) containing X to the total number of records in D. An *association rule* is an implication rule $\Rightarrow Y$, where $X; \subseteq I$ and $X \bigcap Y=0$. The confidence of $\Rightarrow Y$ is the ratio of $\sigma(\bigcup Y/D)$ to $\sigma(X/D)$, indicating that the percentage of those containing X also contain Y. Based on the user-specified minimum support *(minsup)* and confidence *(minconf)*, the following statements are true: An itemset X is frequent if $\sigma(X/D)\geq$ minsup, and an association rule $\Rightarrow XY$ is strong i $\bigcup XY$ is frequent and $\frac{\sigma(X\bigcup Y/D)}{\sigma(X/Y)}$, minconf. The problem of mining association rules is to find all strong association rules, which can be divided into two subproblems:

1. Find all the frequent itemsets.
2. Generate all strong rules from all frequent itemsets.

Because the second subproblem is relatively straightforward — we can solve it by extracting every subset from an itemset and examining the ratio of its support; most of the previous studies (Agrawal, Imielinski, & Swami, 1993; Agrawal, Mannila, Srikant, Toivonen, & Verkamo, 1996; Park, Chen, & Yu, 1995; Savasere, Omiecinski, & Navathe, 1995) emphasized on developing efficient algorithms for the first subproblem.

This article introduces two important techniques for association rule mining: (a) finding N most frequent itemsets and (b) mining multiple-level association rules.

BACKGROUND

An association rule is called *binary association rule* if all items (attributes) in the rule have only two values: 1 (yes) or 0 (no). Mining binary association rules was the first proposed data mining task and was studied most intensively. Centralized on the *Apriori* approach (Agrawal et al., 1993), various algorithms were proposed (Savasere et al., 1995; Shen, 1999; Shen, Liang, & Ng, 1999; Srikant & Agrawal, 1996). Almost all the algorithms observe the downward property that all the subsets of a frequent itemset must also be frequent, with different pruning strategies to reduce the search space. Apriori works by finding frequent k-itemsets from frequent (k-1)-itemsets iteratively for k=1, 2, ..., m-1.

Two alternative approaches, mining on domain partition (Shen, L., Shen, H., & Cheng, 1999) and mining based on knowledge network (Shen, 1999) were proposed. The first approach partitions items suitably into disjoint itemsets, and the second approach maps all records to individual items; both approaches aim to improve the bottleneck of Apriori that requires multiple phases of scans (read) on the database.

Finding all the association rules that satisfy minimal support and confidence is undesirable in many cases for a user's particular requirements. It is therefore necessary to mine association rules more flexibly according to the user's needs. Mining different sets of association rules of a small size for the purpose of predication and classification were proposed (Li, Shen, & Topor, 2001; Li, Shen, & Topor, 2002; Li, Shen, & Topor, 2004; Li, Topor, & Shen, 2002).

MAIN THRUST

Association rule mining can be carried out flexibly to suit different needs. We illustrate this by introducing important techniques to solve two interesting problems.

Finding *N* Most Frequent Itemsets

Give $\subseteq x, yI$, we say that x is greater than y, or y is less than x, if $\sigma(x/D) > \sigma(y/D)$. The largest itemset in D is the itemset that occurs most frequently in D. We want to find the N largest itemsets in D, where N is a user-specified number of interesting itemsets. Because users are usually interested in those itemsets with larger supports, finding N most frequent itemsets is significant, and its solution can be used to generate an appropriate number of interesting itemsets for mining association rules (Shen, L., Shen, H., Pritchard, & Topor, 1998).

We define the rank of itemset x, denoted by $\theta(x)$, as follows: $\theta(x) = |\{\sigma(y/D)>\sigma(x/D), \varnothing\subset y\subseteq I\}|+1$. Call x a winner if $\theta(x)\leq N$ and $\sigma(x/D)\geq 1$, which means that x is one of the N largest itemsets and it occurs in D at least once. We don't regard any itemset with support 0 as a winner, even if it is ranked below N, because we do not need to provide users with an itemset that doesn't occur in D at all.

Use W to denote the set of all winners and call the support of the smallest winner the *critical support*, denoted by *crisup*. Clearly, W exactly contains all itemsets with support exceeding crisup; we also have crisup$\geq$1. It is easy to see that $|W|$ may be different from N: If the number of all itemsets occurring in D is less than N, $|W|$ will be less than N; $|W|$ may also be greater than N, as different itemsets may have the same support. The problem of finding the N largest itemsets is to generate W.

Let x be an itemset. Use $P_k(x)$ to denote the set of all k-subsets (subsets with size k) of x. Use U_k to denote $P_1(I)\cup\ldots\cup P_k(I)$, the set of all itemsets with a size not greater than k. Thus, we introduce the k-rank of x, denoted by $\theta_k(x)$, as follows: $\theta_k(x) = |\{y| \{\sigma(y/D)>x/D), y\in U_k\}|+1$. Call x a k-winner if $\theta_k(x)\leq N$ and $\sigma(x/D)\geq 1$, which means that among all itemsets with a size not greater than k, x is one of the N largest itemsets and also occurs in D at least once. Use W_k to denote the set of all k-winners. We define *k-critical-support*, denoted by *k-crisup*, as follows: If $|W_k|<N$, then k-crisup is 1; otherwise, k-crisup is the support of the smallest k-winner. Clearly, W_k exactly contains all itemsets with a size not greater than k and support not less than k-crisup. We present some useful properties of the preceding concepts as follows.

Property: Let k and i be integers such that $1\leq k\leq k+i \leq |I|$.

(1)Given $x\in U_k$, *we have* $x\in W_k$ iff $\sigma(x/D)\geq k$-crisup.
(2) If $W_{k-1} = W_k$, then $W=W_k$
(3)$W_{k+i}\bigcap U_k\subseteq W_k$.
(4) $1\leq k$-crisup$\leq(k+i)$-crisup.

To find all the winners, the algorithm makes multiple passes over the data. In the first pass, we count the supports of all 1-itemsets, select the N largest ones from them to form W_1, and then use W_1 to generate potential 2-winners with size (2). Each subsequent pass k involves three steps: First, we count the support for potential k-winners with size k (called candidates) during the pass over D; then select the N largest ones from a pool precisely containing supports of all these candidates and all (k-1)-winners to form W_k; finally, use W_k to generate potential (k+1)-winners with size k+1, which will be used in the next pass. This process continues until we can't get any potential (k+1)-winners with size k+1, which implies that $W_k+1 = W_k$. From Property 2, we know that the last W_k exactly contains all winners.

We assume that M_k is the number of itemsets with support equal to k-crisup and a size not greater than k, where $1\leq k\leq |I|$, and M is the maximum of all $M_1\sim M_{|I|}$. Thus, we have $|W_k|=N+M_k-1<N+M$. It was shown that the time complexity of the algorithm is proportional to the number of all the candidates generated in the algorithm, which is $O(\log(N+M)*min\{N+M,|I|\}*(N+M))$ (Shen et al., 1998). Hence, the time complexity of the algorithm is polynomial for bounded N and M.

Mining Multiple-Level Association Rules

Although most previous research emphasized mining association rules at a single concept level (Agrawal et al., 1993; Agrawal et al., 1996; Park et al., 1995; Savasere et al., 1995; Srikant & Agrawal, 1996), some techniques were also proposed to mine rules at generalized abstract (multiple) levels (Han & Fu, 1995). However, they can only find multiple-level rules in a fixed concept hierarchy. Our study in this fold is motivated by the goal of