Towards Efficient Data Re-mining (DRM)

Jiming Liu¹ and Jian Yin²

¹Department of Computer Science, Hong Kong Baptist University
Kowloon Tong, Hong Kong jiming@comp.hkbu.edu.hk

²Department of Computer Science, Zhongshan University
Guangzhou, 510275, P. R. China issjyin@zsu.edu.cn

Abstract. The problem that we tackle here is a practical one: When users interactively mine association rules, it is often the case that they have to continuously tune two thresholds: minimum support and minimum confidence, which describe the users’ changing requirements. In this paper, we present an efficient data re-mining (DRM) technique for updating previously discovered association rules in light of threshold changes.

1 Introduction

Various algorithms have been proposed [1,2,4,6] to discover frequent item-sets. Generally speaking, these algorithms first construct a candidate set of frequent item-sets based on certain heuristics, and then discover the subset that indeed contains frequent item-sets. This process can be done iteratively in the sense that the frequent item-sets discovered at one iteration will be used as the basis for generating the candidate set for the next iteration. For example, in [2], at the kth iteration, all frequent item-sets containing k items, referred to as frequent k-item-sets, are generated. In the next iteration, to construct a candidate set of frequent (k+1)-item-sets, a heuristic is used to expand some frequent k-item-sets into a (k+1)-item-set, if certain constraints are satisfied.

Among all the algorithms proposed, Apriori (and its variants) [2] and DHP [3] algorithms are most commonly applied. They both run a number of iterations and compute the frequent item-sets of the same size at each iteration, starting from the size-one item-sets. At each iteration, they first construct a set of candidate item-sets and then scan the database to count the number of transactions that contain each candidate set. The key to optimization lies in the techniques used to create the candidate sets. The smaller the number of candidate sets is, the faster the algorithms would be.

However, very little work has been done on the second problem mentioned earlier. A method of handling incremental database updates for the rules discovered by the generalization-based approach was briefly discussed in [5]. As related to this problem, Lee and Cheung have done some work [7], which focuses primarily on how to update association rules when a database is incrementally changed. As in real-world applications, users are often unsure about their requirements on the minimum support and confidence in the first place. This can be due to the lack of knowledge about the application domains or the outcomes resulting from different threshold settings. As a result,
they may be repeatedly unsatisfied with the association rules discovered, and hence
need to re-execute the mining procedure many times with varied thresholds. In the
cases where large databases are involved, this could be a time-consuming, trial-and-
error process, since all the computation done initially in finding the old frequent item-
sets is wasted and all frequent item-sets have to be re-computed again from scratch. In
order to deal with this situation, it is both desirable and imperative to develop an effi-
cient means for **re-mining a database under different thresholds** in order to obtain
an acceptable set of association rules. In this paper, we will explicitly address this
problem and present an efficient algorithm, called Posteriori, for computing the fre-
quent item-sets under the varied thresholds.

## 2 Problem Statement

### 2.1 Mining Association Rules

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of literals, called items. Let $D$ be a set of transactions, where
each transaction $T$ is a set of items such that $T \subseteq I$. Associated with each transaction is
a unique identifier, called its TID. We say that a transaction $T$ contains $X$, a set of
some items in $I$, if $X \subseteq T$. An association rule is an implication of the form $X \Rightarrow Y$,
where $X \subseteq I$, $Y \subseteq I$, and $X \cap Y = \emptyset$. The rule $X \Rightarrow Y$ holds in the transaction set $D$ with
confidence $c$ if $c \%$ of transactions in $D$ that contain $X$ also contain $Y$. The rule $X \Rightarrow Y$
has support $s$ in the transaction set $D$ if $s \%$ of transactions in $D$ contain $X \cup Y$.

### 2.2 Re-mining Association Rules

Let $L$ be the set of frequent item-sets in $D$, $s$ be the minimum support, and $|D|$ be the
number of transactions in $D$. Assume that for each $X \in L$, its support count, $X\text{.support}$,
which is the number of transactions in $D$ containing $X$, is available.

After users have found some association rules, they may be unsatisfied with the re-
sults and want to try out new results with certain changes on thresholds, such as min-
sup from $s$ to $s'$.

Thus, the essence of the problem of re-mining association rules is to find the set $L'$
of frequent item-sets under the new thresholds. Note that a frequent item-set in $L$
may not be a frequent item-set in $L'$. On the other hand, an item-set $X$ not in $L$ may become
a frequent item-set in $L'$.

## 3 Algorithm Posteriori

The following notations are used in the rest of the paper. $L_k$ is the set of all size-$k$
frequent items in $D$ under the support of $s\%$, and $L_k'$ is the set of all frequent k-items
in $D$ under the support of $s'\%$. $C_k$ is the set of of size-k candidate sets in the k-th itera-