CS 6604: Data Mining Fall 2007

Lecture 6 — September 5, 2007

Lecture: Naren Ramakrishnan *Scribe: Jae-seung Yeom*

1 Overview

In this lecture we introduce a set of deduction rules, which are based on the inclusion-exclusion principle, to derive bounds on the support of a candidate itemset, given the supports of all its subsets. Then, we discuss possible performance improvements in frequent itemset generation by using these deduction rules and trying to avoid scanning the database.

2 Using the Inclusion-Exclusion Principle

2.1 Inclusion-exclusion principle

Let $A_1, \ldots, A_n$ be n finite sets. Then, the inclusion-exclusion principle is the following.

$$|\bigcup_{i=1}^{n} A_i| = \sum_{i \le i \le n} |A_i| - \sum_{1 \le i < j \le n} |A_i \cap A_j| + \cdots - (-1)^n |\bigcap_{i=1}^{n} A_i|$$

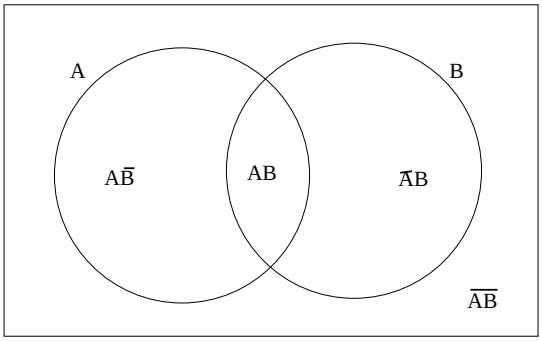
or

$$\text{size}(A_1 \cup A_2 \cdots \cup A_n) = \sum_{X \subset A} (-1)^{|X|-1} \text{size}(X),$$

where $A = A_1 \cup A_2 \cdots \cup A_n$

2.2 Bounds for supports

From the Venn diagram above, we can write:



$$
\begin{aligned}
support(A\overline{B}) &= support(A) - support(AB) \\
support(\overline{A}B) &= support(B) - support(AB) \\
support(\overline{AB}) &= N - support(A) - support(B) + support(AB) \\
support(AB) &= support(AB) \\
N = support(\emptyset)
\end{aligned}
$$

We can think of the left sides of the above expressions as 'generalized itemsets.' So $support(A\overline{B})$ refers to those transactions that contain A but not B. Any of these itemsets has $support \geq 0$. This gives:

$$
\begin{aligned}
support(A) - support(AB) \geq 0 \\
support(B) - support(AB) \geq 0 \\
support(\emptyset) - support(A) - support(B) + support(AB) \geq 0
\end{aligned}
$$

Then, we have the following rules which are universally true.

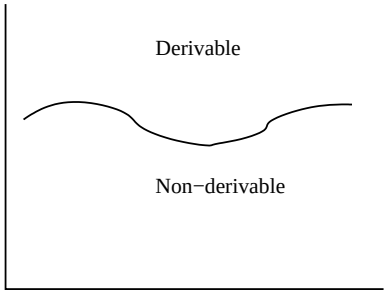
$$
\begin{aligned}
support(AB) &\leq support(A) \\
support(AB) &\leq support(B) \\
support(AB) &\geq support(A) + support(B) - support(\emptyset) \\
support(AB) &\geq 0
\end{aligned}
$$

There are two '$\geq$'s and two '$\leq$'s. We can use these rules to get the bounds of $support(AB)$ from $support(A)$ and $support(B)$. Thus, the support bounds of a level 2 itemset is obtained from the (actual) supports of level 1 itemsets. For The tightest bounds, we use the highest lower bound and lowest upper bound. None of these rules are redundant [1]. Therefore, omission of the rule will result in a less tight interval. Bounds may be useful to check whether the support of a level 2 itemset would be above the minimum support, without actually counting it. Similarly, support bounds of level 3 itemsets can be derived from the supports of level 2 and level 1 itemsets as well, as discussed in [1].

2.3 Bounds and minimum support

How can we use these bounds to prune our search space? Given a bound $[x, y]$ on the support of an itemset the following cases can happen:

1. case 1: if $[x, y]$ and $y < min\_support \Rightarrow$ infrequent



2. case 2: if $[x, y]$ and $x \geq min\_support \Rightarrow$ frequent

3. case 3: if $x = y$ in $[x, y]$
   $\Rightarrow$ frequent (if $x \geq min\_support$)
   $\Rightarrow$ infrequent (if $x < min\_support$)

Those itemsets that belong to the last case, i.e., where the lower and upper bounds are equal, are called *derivable* itemsets. All other itemsets are called non-derivable itemsets. Derivability is monotone. There is a border between derivable and non-derivable itemsets. Hence, when searching from bottom to top, the moment we reach the derivable boundary, we do not need to search itemsets beyond that point.

3 Constraints

We now move on to finding itemsets with constraints. We will merely state some useful constraints here and think about algorithms for mining them in next class.

1. Minimum support constraint: anti-monotonic

2. "itemset with avg(price) $\leq$ \$4": this is nasty nasty because it requires us to sum up attributes of itemsets. It is neither anti-monotonic or monotonic

3. An even nastier constraint involves assessment of variance of an itemset's attribute.

4. Succinct constraint: to be continued in next class

Monotonic: Given $X \subseteq Y$, if X has the property, Y has the property
Anti-monotonic: Given $X \subseteq Y$, if X does not have the property, Y does not have the property.

References

[1] T. Calders and B. Goethals, "Mining all Non-Derivable Itemsets," *Data Mining and Knowledge Discovery*, 2006

[2] J. Pei and J. Han, "Can we push more Constraints into Frequent Pattern Mining?" *in Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 350-354, 2000.