EX04

Textbook (3rd Edition)

- 6.1, 6.3, 6.4, 6.5, 6.6, 6.11

Due on 3/3

6.1 Suppose you have the set C of all frequent closed itemsets on a data set D, as well as the support count for each frequent closed itemset. Describe an algorithm to determine whether a given itemset X is frequent or not, and the support of X if it is frequent.

Algorithm called: Frequent_determinator.

Input:

The set C of all frequent closed itemsets on a dataset D.

The support count for each frequent closed itemset in C.

An itemset X whose frequency and support count you want to determine.

Output:

Boolean value indicating whether the given itemset X is frequent or not.

The support count of X (if X is frequent)

Function:

```
function Frequent_determinator (X, C)
// Initialize s as an empty set
s = ∅

// Iterate over each itemset l in C
for each itemset l in C
    if X ⊂ l and (s = ∅ or length(l) < length(s))
        s = l

// If s is not empty return the support count of s
if s ≠ ∅
    return supportCount(s)

// Else return -1
return -1
```

6.3 The Apriori algorithm makes use of prior knowledge of subset support properties.
(a) Prove that all nonempty subsets of a frequent itemset must also be frequent.
(b) Prove that the support of any nonempty subset *s* of itemset *s* must be at least as great as the support of *s*.
(c) Given frequent itemset l and subset s of l, prove that the confidence of the rule “s′ ⇒(l−s′)”cannot be more than the confidence of “ s⇒(l−s), ” where s′ is a subset of s.
(d) A partitioning variation of Apriori subdivides the transactions of a database D into n nonoverlapping partitions. Prove that any itemset that is frequent in D must be frequent in at least one partition of D.

(a).
Let $S \subseteq I$ be a frequent itemset and let $\emptyset \neq S' \subseteq S$. Thus, support(S) ≥ minSup. Then support(S′) ≥ support(S), support(S′) ≥ minSup. Thus, S is a frequent itemset. All nonempty subsets of a frequent itemset must also be frequent.

(b)
Let $\emptyset \neq S' \subseteq S \subseteq I$. For any transaction $T \subseteq I$ in database D, we have $S \subseteq T \Rightarrow S' \subseteq T$. Thus, $\{T \in D \mid S \subseteq T\} \subseteq \{T \in D \mid S' \subseteq T\}$. In the end, support(S) = $\{T \in D \mid S \subseteq T\} \leq \{T \in D \mid S' \subseteq T\}$ = support (S′)

(c)
First, we assume there are frequent items l and its subsets s and s′, where s′ is a subset of s. The confidence of the rule s′ ⇒(l − s′) cannot be more than the confidence of s ⇒ (l − s). The confidence of s ⇒ (l − s) is calculated as: Conf(s ⇒ (l − s)) = Support(s ∪ (l − s))/ Support(s). Since s ∪ (l − s) = l, this simplifies to Support(l) / Support(s). Similarly, the confidence of s′ z⇒ (l − s′) is Support(l) / Support(s’). In part b, we know that Support(s) ≥ Support(s′), thus confidence s ⇒(l − s) ≥ s′ ⇒(l − s′). Therefore, we have shown that the confidence of the rule s′ ⇒(l − s′) cannot be more than the confidence of s⇒(l − s).

(d)
Assume the itemset is not frequent in any partition.

Let F be a frequent itemset in database D, where D contains numerous transaction records. Let C be the total number of transactions in D, and A be the number of transactions containing F, with min sup being the minimum support. Since F is frequent, A = C × min sup. D is divided into n non-overlapping partitions d1, d2, d3, ..., dn, with C being the sum of transactions across all partitions, and A being the sum of transactions containing F across all partitions.

Based on the assumption that F is not frequent in any partition, it means the number of transactions containing F in each partition is less than the number of transactions in that partition multiplied by min sup. Applying this logic across all partitions, we find that $A < C \times$ min sup, contradicting the definition of F as a frequent itemset.

Therefore, the assumption is incorrect, indicating that a frequent itemset in database D must be frequent in at least one of its partitions.

6.4 Let *c* be a candidate itemset in C_k generated by the Apriori algorithm. How many length-($k - 1$) subsets do we need to check in the prune step? Per your previous answer, can you give an improved version of procedure has infrequent subset in Figure 6.4?

In the pruning step, we must check k length–k − 1 subsets for the candidate itemset c generated by the Apriori algorithm. However, since c is formed by merging two length-k − 1 frequent itemsets, we only need to examine k−2 length-k − 1 subsets, excluding the two subsets we already know are frequent.

6.5 Section 6.2.2 describes a method for generating association rules from frequent itemsets. Propose a more efficient method. Explain why it is more efficient than the one proposed there. (Hint: Consider incorporating the properties of Exercises 6.3(b), (c) into your design.)

```
Algorithm: Efficient_Association_Rule_Generation
Input:
    Frequent Itemsets F, Minimum Confidence Threshold minConf
Output:
    Valid Association Rules R
Function:
R = [ ]

for f in F:
    for s in non_empty_subsets(f):
        consequent = f - s
        confidence = calculate_confidence(s, consequent)
        if confidence >= minConf:
            R.append((s, consequent, confidence))
            for i in items_not_in_f(f):
                if decreases_confidence(s, consequent.union({i}), minConf):
                    continue

return R
```

This method achieves selective rule generation by only considering the non-empty subsets of frequent itemsets for potential rule generation, avoiding the inefficiency of generating all possible rules and then filtering them based on confidence. This strategy significantly reduces the number of rules that need to be evaluated, enhancing overall efficiency. Moreover, the method calculates confidence only when necessary, ensuring all generated rules are relevant and potentially valuable without the need to reassess the support for each potential rule's antecedent and consequent. This efficient approach to confidence calculation further optimizes the algorithm's performance, making it more efficient and effective in mining association rules.

6.6 A database has five transactions. Let min sup = 60% and min conf = 80%.

(a) Find all frequent itemsets using Apriori and FP-growth, respectively. Compare the efficiency of the two mining processes.

i. Using the Apriori algorithm, the following frequent itemsets and candidate itemsets were identified (after pruning based on the presence of infrequent subsets):

L1 includes: {E, K, M, O, Y}
C2 includes: {EK, EM, EO, EY, KM, KO, KY, MO, MY, OY}
L2 includes: {EK, EO, KM, KO, KY}
C3 includes: {EKO}
L3 includes: {EKO}
C4 is empty: ∅
L4 is also empty: ∅
This leads to the complete set of frequent itemsets being: {E, K, M, O, Y, EK, EO, KM, KO, KY, EKO}

ii. Using the FP-growth algorithm, the frequent itemsets are discovered as follows:

The list of single frequent items (L) with their support counts is given by: {E: 4}, {K: 4}, {M: 3}, {O: 3}, {Y: 3}.
For the item Y, the CPB is composed of: {E, K, M, O: 1}, {E, K, O: 1}, {K, M: 1}. The CFPT for Y is ⟨K: 3⟩. This generates the frequent pattern {K, Y} with a support count of 3.
For the item O, the CPB includes:{E, K, M: 1}, {E, K: 2}. The CFPT for O is ⟨E: 3, K: 3⟩. This leads to the generation of the following frequent patterns: {E, K, O} with a support count of 3, {K, O} with a support count of 3, and {E, O} with a support count of 3.
For the item M, the CPB is: {E, K: 2}, {K: 1}. The CFPT for M is: ⟨K: 3⟩ This results in the frequent pattern {K, M} with a support count 3.
For the item K, the CPB shows:{E: 4}. The CFPT for K is: ⟨E: 4⟩ This generates the frequent pattern {E, K} with a support count 4.
In conclusion, the complete set of frequent itemsets identified through FP-growth includes: { {E: 4}, {K: 4}, {M: 3}, {O: 3}, {Y: 3}, {K, Y: 3}, {E, K, O: 3}, {K, O: 3}, {E, O: 3}, {K, M:3}, {E, K: 4} }

FP-growth is more efficient because it directly mines the conditional pattern bases, which can significantly reduce the size of data sets to be searched, especially as the dataset grows. However, for small datasets like the provided example, particularly when processed manually, one might perceive Apriori as more "efficient" due to its simplicity and the tangible progression through the itemset lattice.

(b) List all the strong association rules (with support s and confidence c) matching the following metarule, where X is a variable representing customers, and itemi denotes variables representing items (e.g., “A,” “B,”):

$\forall x \in$ transaction, buys(X, item1) $\wedge$ buys(X, item2) $\Rightarrow$ buys(X, item3)

$\forall x \in$ transaction, buys(X, E) $\wedge$ buys(X, O) $\rightarrow$ buys(X, K) (60%, 100%) Strong
$\forall x \in$ transaction, buys(X, K) $\wedge$ buys(X, O) $\rightarrow$ buys(X, E) (60%, 100%) Strong

6.11 Most frequent pattern mining algorithms consider only distinct items in a transaction. However, multiple occurrences of an item in the same shopping basket, such as four cakes and three jugs of milk, can be important in transactional data analysis. How can one mine frequent itemsets efficiently considering multiple occurrences of items? Propose modifications to the well-known algorithms, such as Apriori and FP-growth, to adapt to such a situation.

When faced with transaction data where items may appear multiple times within the same basket, such as four cakes and three jugs of milk, traditional frequent pattern mining algorithms require adjustments to mine frequent item sets effectively. For the Apriori algorithm, one approach is to treat each item occurrence based on its count value as a separate item, meaning an item appears twice. It once would be considered two distinct items like "A:1" and "A:2". This allows for the generation of frequent 2-itemsets and 3-itemsets ..., by considering the count values of items while maintaining the original logic of the algorithm and checking these itemsets against the minimum support requirement. For the FP-growth algorithm, it is also necessary to consider the count values of items when constructing the frequent pattern tree (FP-tree) and to treat each item associated with different count values as different paths when projecting the item databases. Moreover, further optimization, such as strategies for mining closed item sets, can enhance the efficiency of the algorithms in processing transactions that include repeated items. These modifications enable Apriori and FP-growth algorithms to handle more complex data scenarios and enhance the depth and breadth of data analysis, revealing more detailed consumption patterns and preferences.