Yangxin Fan

CSC 440 Data Mining . HW#3

6.1 Algorithm for determining frequent itemset and its support given a set of close frequent itemsets: Countlist = [ ]

```
if X == ∅:
    print ("not frequent itemset")
for each frequent closed itemset f ∈ C:
    if X ⊂ f:
        countlist.append [support(f)]
return max(Countlist)
if countlist == [ ]
    print ("not frequent itemset")
```

Note that: the support(f) corresponds to max in countlist is the one with shortest length. The algorithm return the support of X if X is a frequent itemset, otherwise just print "not frequent itemset."

6.3

(a). Prove all non-empty subsets of a frequent itemset must be frequent (Proof)

Let A be a frequent itemset, min_sup be the minimum support. Let D be dataset, |D| be number of all transactions, $A' \subset A$, any transaction contains A also have A'.

We have support_count(A) $\geq$ min_sup $\times$ |D|

since $A' \subset A$ so support_count(A') $\geq$ support_count(A) $\geq$ min_sup $\times$ |D|

Hence, A' the subset of frequent itemset A must also be frequent.

(b). Since any transactions include S must also include S' where S'⊂S but not the other way around. ⇒ Support_count(S') ≥ Support_count(S) ⇒ Support(S') ≥ Support(S)
This proves part (b).

(c).
Since Confidence(S'⇒(I-S')) = $\frac{\text{Support}(I)}{\text{Support}(S')}$
Also confidence(S⇒(I-S)) = $\frac{\text{Support}(I)}{\text{Support}(S)}$
From part b, we know since S'⊂S, ⇒ support(S') ≥ support(S)
Hence. confidence(S'⇒(I-S')) ≤ confidence(S⇒(I-S))

(d). Since we know an itemset A is frequent in D. so(*) Support_count(A) ≥ min_sup × |D|, where min_sup is minimal support, |D| is total # transactions of dataset D.
If we divide D into n non-overlapping dataset $D_1, \ldots, D_n$
(Δ) Assume that A is not frequent in any of partition of D. ⇒ $\frac{\text{Support\_count}(A)_{Di}}{|Di|}$ < min_sup for i ∈ {1, ..., n}

$\Rightarrow$ Support-count$(A)_{D_i} < \text{min_sup} \times |D_i|$
for all $i \in \{1, \ldots, n\}$

$\Rightarrow$ Support-count$(A)_{D_1} + \ldots + \text{Support-count}(A)_{D_n} = \text{Support-count}(A)$
$< \text{min\_sup} \times \sum_i |D_i| = \text{min\_sup} \times |D|$

$\therefore$ This contradicts (*) support_count(A) $\geq$ min_sup $\times |D|$ so our assumption (Δ) is wrong

Hence, A is frequent in at least one partition of D. (Proof by contradiction)

6.4. We actually only need to check (k-2) subsets of length (k-1) of C. Since we know C is generated from (k-1) itemset l_1, l_2, there's no need to check frequent whether l_1, l_2 are in L_{k-1}. One way to do this is to pass l_1, l_2 as arguments to function has_infrequent_subset.

6.5. Since we know $\text{Conf}(S' \Rightarrow l-S') \leq \text{Conf}(S \Rightarrow l-S)$, if $S' \subset S$. We can improve association rule generation. If we already know a subset of s of l shows $\text{Conf}(S \Rightarrow l-S) < \text{min\_conf}$, then there's no need to check all the non-empty subsets of S. In this way, we don't need to check every subset of l whether each of them has conf $\geq$ min_sup.

Algorithm: Recursion

Input: L frequent itemset, minimal confidence (min_conf)

Output: itemsets in L with confidence ≥ min_conf.

for each frequent itemset l
    generating_itemset_association(l, L, min_conf)

Function generating_itemset_association(C, L): C: current subset of L, L: original frequent itemset, min_conf).

K = length(C)
if (K>1) then
    Generate all (K-1) length subsets of L
    for each (K-1) subset m of L
        if (support(L)/support(m) ≥ min_conf)
            print ("m ⇒ (L-x)")
            generating_itemset_association(m, L, min_conf)

This is much more efficient. we don't need to check every subset of L, in fact, we only check those whose superset S has (S ⇒ L-S).

6.6. min_sup = 60%, min_conf = 80%

(a). Part I: find frequent itemset using Apriori.

∵ min_sup = 60% ∴ min_support_count = 3.

$L_1 = \{M, O, K, E, Y\}$

$C_2 = \{MO, MK, ME, MY, OK, OE, OY, KE, KY, EY\}$

$L_2 = \{MK, OK, OE, KE, KY\}$

$C_3 = \{OKE\}$

$L_3 = \{OKE\}$

$C_4 = \emptyset$

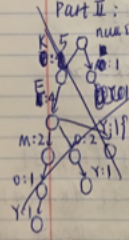
$L_4 = \emptyset$

So the set of frequent itemsets is:

$\{M, O, K, E, Y, MK, OK, OE, KE, KY, OKE\}$

Part II: find frequent ~~itemset using~~ FP-growth

nuest $L = \{\{K:5\}, \{E:4\}, \{M:3\}, \{O:3\}, \{Y:3\}\}$

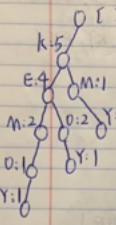


Item = Y

Conditional pattern base = $\{\{E,K,M,O:1\}, \{E,K,O:1\}\}$

Part I. find frequent itemset use FP-Growth.

L = {{K:5}, {E:4}, {M:3}, {O:3}, {Y:3}}



Item = Y

Conditional pattern base = {{K,E,M,O:1}, {K,E,O:1}, {K,M:1}}.

Conditional FP tree = <K:3>.

Frequent pattern: {K,Y:3}.

Item = O

CPB = {{K,E:2}, {K,E,M:1}}

CFPT = <K:3, E:3>

FP: {K,O:3}, {K,E,O:3}, {E,O:3}.

Item = M, CPB = {{K,E:2}, {K:1}}

CFPT = <K:3>, FP: {K,M:3}

Item = E, CPB = {K:4} CFPT = <K:4>.

FP: {K,E:4}

Hence, the frequent itemset is

{{K:5}, {E:4}, {M:3}, {O:3}, {Y:3}, {K,Y:3},

{K,O:3}, {E,O:3}, {K,E,O:3}, {K,M:3}, {K,E:4}}.

Summary: FP-growth is much more efficient since it only needs to scan database twice and we don't need to generate candidate sets frequently.

(b).

Let's take a look at the frequent itemset we found in part (a). Note: we only need to consider itemset with ~~two~~ three items

Start from itemset with three items $\{K, E, O: 3\}$

We found. $\forall x \in$ transaction $\textcircled{1}$ $buys(X, O) \wedge buys(X, E) \Rightarrow buys(X, K)$ [60%, 100%]

$\textcircled{2}$. $buys(X, O) \wedge buys(X, K) \Rightarrow buys(X, E)$ [60%, 100%]

~~for itemset with two items,~~

~~we found. $\forall x \in$ transaction, $buys(X, K) \Rightarrow buys(X, E)$ [80%,~~

$\textcircled{3}$. $buys(X, E) \wedge buys(X, K) \Rightarrow buys(X, O)$ [60%, 75%]

Hence, $\textcircled{1}$, $\textcircled{2}$ are strong association matches requirement (form).

6.11.

When extending to Apriori and FP Growth by considering multiple occurences of items. we need to treat ~~same~~ every item with different counts as different items. In our first scan of database for both methods, we will generate frequent single itemset by taking account to different counts. Then check if the minimal support is met. For Apriori, we can continue to

Construct frequent 2-itemsets, 3-itemsets.
For FP-growth when we project database.
We also have to project bth itemsets where each item can be associated with different counts.