Meiying Chen

DSC 440: Data Mining

October 3, 2019

Homework 3

Problem Set 6.1

Because we are not sure if X is closed or not, but C is a set of closed frequent itemsets, so we can figure out whether the subsets of X is closed and belong to C. We explore from the longest subsets, which is X, and the sequential subsets is one item shorter.

Algorithm: determine whether a given item set X is frequent or not

Input:

- C: all frequent closed itemsets o data set D and their support count
- X: the given itemset

Output:

- isfrequent: true if X is frequent, false if not
- s: support count of X if it is frequent

Method:

```
function check_frequent(C, X):
  isfrequent = false
  s = -1
  for each subset of X:
    from the largest to the smallest sub set x_sub:
      if x_sub is in C:
        isfrequent = true
        s = frequent(s_sub)
      end
  end
  return isfrequent, s
```

Problem Set 6.3

(a) If an itemset is nonempty and not frequent, then no matter what element(item) is added, the super itemset will not be frequent. That is to say, non-frequent subsets cannot form a frequent subsets. So nonempty subsets of a frequent itemset must be frequent.

(b) Given a nonempty subset s' and its support sup(s'), assume that it contains k items less than itemset s, which has support sup(s).

$$
\begin{gathered}
because: sup(s') \geq sup(s'+1) \\
then: sup(s') \geq sup(s'+(1...k)) \\
equalto: sup(s') \geq sup(s)
\end{gathered}
$$

(c)

$$
confidence(s \Rightarrow (l-s)) = P((l-s)|s) = \frac{sup(s \cup (l-s))}{sup(s)} = \frac{sup(l)}{sup(s)} \tag{1}
$$

$$
confidence(s' \Rightarrow (l-s')) = P((l-s')|s') = \frac{sup(s' \cup (l-s'))}{sup(s')} = \frac{sup(l)}{sup(s')} \tag{2}
$$

$$
because: sup(s') \geq sup(s)
$$

$$
so: euqation(1) \leq euqation(2)
$$

(d)

$$
\begin{gathered}
\forall i \subseteq D, \exists i \subseteq S \\
assume \ \forall s \subseteq D, s \ is \ not \ frequent \\
so \ for \ s(i) \ that \ i \subseteq s(i): \frac{sup(i)}{sup(s(i))} < minsup(s(i)) \\
if \ we \ stack \ all \ s \ together, \ then: \frac{sup(i)}{sup(D)} < minsup(D) \\
but \ \forall i \subseteq D, i \ is \ frequent: \ \frac{sup(i)}{sup(D)} \geq minsup(D), which \ disagrees \ with \ the \ former \ euqation \\
so, \ the \ assumption \ is \ rejected
\end{gathered}
$$

Problem Set 6.4

If annotate the length of c as len(c), then the subsets with length k-1 we need to check is:

$$C_{len(c+1)}^{len(c)}$$

Possible improvement of the has_infrequenct_subset function:

```
procedure has_infrequent_subset(c, L_(k-1));
  for each item in c:
    if any c not in L_(k-1):
      return false;
    else:
      for each k-1 subset s of c:
        if s not in L_(k-1): return false;
      return true;
  return true;
```

Problem Set 6.5

Algorithm:

```
from k = 1 to k = length(l)/2:
   init set seed;
   for each subset s' of length k:
    if s' is confident: add s' to seed;
   end
end
form longer subsets using combinations of elements in seed
```

Explanation:

Because we know from the question 6.3(c):

$$if\ s'\ is\ the\ subset\ of\ s, then: confidence(s \Rightarrow (l-s)) \geq confidence(s' \Rightarrow (l-s'))$$

Because we are looking for subsets s that makes confidence(s⇒(l-s)) ⩾ min_confidence. So if a subset s' of s suffices such a condition, then super set of s' will suffice the condition too. So instead of checking all subsets of s, we can start from looking for the short subset s' which confidence surpass the required minimum confidence, and form the longer sets using this shorter subsets.

Problem Set 6.6

(a) **Apriori:**

min_sup = 5*60% = 3

C1:

Item	Support
K	5
E	4
O	4
M	3
Y	3
C	2
N	2
A	1
D	1
U	1
I	1

L1:

Item	Support
K	5
E	4

O	4
M	3
Y	3

C2:

Item	Support
{K,E}	4
{K,O}	3
{K,M}	3
{K,Y}	3
{E,O}	3
{E,M}	2
{E,Y}	2
{O,M}	1
{O,Y}	2
{M,Y}	2

L2:

Item	Support

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

C3:

Item	Support
{K,E,O}	3
{K,E,M}	2
{K,E,Y}	2
{K,O,M}	1
{K,O,Y}	2
{K,M,Y}	2
{E,O,Y}	2
{E,O,M}	1

L3:

Item	Support

{K,E,O}	3	

No further combination can be made, so the longest frequent itemset has 3 items.

FP-growth

Support count sort:

K 5> E 4 = O 4> M 3 = Y 3 > C 2 = N 2 > A 1 = D 1 = U 1 = I 1

Item	Conditional Pattern Base	Conditional FP Tree	Frequent Pattern
O	{KEM:1}, {KE:2}	{K:3}, {E:3}	{KO:3},{EO:3}, {OKE:3}
E	{K:4}	{K:4}	{KE:4}
Y	{KEMO:1},{KEO:1}, {KM:1}	{K:3}	{KY:3}
M	{KE:2},{K:1}	{K:3}	{MK:3}

The Apriori algorithm is apparently slower that FP-growth during the calculating stage as it goes through all dataset several times. However, Apriori algorithm outputs all frequent itemsets results as it stops exploring the dataset, and FP-growth need an extra stage of forming them. But the FP-growth is way better than Apriori algorithm in general.

(2) We can infer from above:

{O,K} ⇒ E, confidence = 100%

{O,E} ⇒ K, confidence = 100%

In required form is:

$$\forall x \in transaction, buys(O) \wedge buys(K) \Rightarrow buys(E)$$
$$\forall x \in transaction, buys(O) \wedge buys(E) \Rightarrow buys(k)$$

Problem Set 6.11

We can consider the frequencies of one item selected in a same basket and the different item count together in the first stage of the FP-growth algorithm. The first step of the FP-growth needs a sort of the support count. To include the frequency attribute, we can define the sorting standard as considering support count and frequency together. For example, item A and item B has same support account, but item A outnumbers item B in the average frequency appearing in one basket. In this situation, we can sort A before B in the first step.