Yuan Wang

440 Data Mining

Prof. Jiebo Luo

Homework 3

**6.1**

**Algorithm:** Determine if an itemset is frequent.

**Input:** *C*, set of all frequent closed itemset along with their support counts; test itemset, *X*.

**Output:** Support of X if it is frequent, otherwise -1.

**Method:**

s = none

for l in C:

if X in l and len(l) < len(s) or s = none:

s=l

if s =! none:

return support(s)

else:

return -1

**6.3**

**a.**

Suppose *s* is the frequent itemset, *min\_sup* is the minimum support. *D* is task relevant data, a set of data transactions. And is the number of transactions of *D*.

So, *support\_count(s) min\_sup*

If *s’* is a nonempty subset of *s*. So, any transaction itemset contains *s* will also contain itemset *s’*. Thus, *support\_count(s’) support\_count(s) min\_sup .*

Therefore, *s*’ is a frequent itemset as well.

**b.**

From question (a) we know that *support\_count(s’) support\_count(s)*, so

Support (s’) = Support (s) = . Therefore, the support of any nonempty subset s ′ of itemset *s* must be as great as the support of *s*.

**c.**

*s* is a subset of *l*, then confidence (*s (l - s)) =*

*s’* is a subset of s, then confidence *(s’ (l – s’)) =*

Because *support (s’) Support (s’),* so *confidence (s’ (l – s’)) confidence (s (l - s)).* Therefore, the conﬁdence of the rule “*s’ ⇒(l – s’ )*” cannot be more than the conﬁdence of the rule “*s⇒(l − s)”*.

**d.**

Proof by Contradiction: Assume that the itemset is not frequent in any of the partitions of D.

Suppose *F* is any frequent itemset. *D* is task relevant data, a set of data transactions. *C* is the total number of transactions in *D. A* is the total number of transactions in *D* containing the itemset F.

So, *A = C min\_sup*.

In the beginning, we suppose *F* is not frequent in any of partitions of *D*. So, *A C × min\_sup*.

This contradicts with what we defined that *F* is frequent itemset.

Therefore, any itemset that is frequent in *D* must be frequent in at least one partition of *D*.

**6.4**

Because is generated from 2 itemsets from , so these 2 subsets of should not need to check. You only need to check the rest of *length-(k − 1)* subsets of , that is total of *k-2* subsets.

One possible improvement is to pass and , and prevent searching for these two subsets because they are frequent itemsets.

**6.5**

The method in Section 6.2.2 generates all the nonempty subsets of a frequent itemset *l* and then tests all of them for potential rules. It may generate and test many unnecessary subsets. The proposed method as below only generates and tests the necessary subsets.

* If a subset *x* of length *k* does not meet the minimum conﬁdence, then there is no need to generate any of its nonempty subsets as their respective conﬁdences will never be greater than the conﬁdence of *x*. (6.3b)
* If *x* meets the minimum conﬁdence then we generate and test its *(k − 1)-subsets*. It will start with the *(n − 1)-subsets* of an n-itemset and progressively work our way down to the *1-subsets*.

**6.6**

**a.**

**Graphical user interface, application, table, Excel

Description automatically generated**

**Apriori:**

= { E, K, M, O, Y } #after delete infrequent subset

= { EK, EM, EO, EY, KM, KO, KY, MO, MY, OY }

= { EK, EO, KM, KO, KY }

= { EKO }

= { EKO}

C4 = ∅

L4 = ∅

Results of frequent itemsets: { E, K, M, O, Y, EK, EO, KM, KO, KY, EKO }

**FP-growth**





Results of frequent itemsets:

{ { 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 } }

Because FP-growth can mine in the conditional pattern bases, it reduces the size of the data sets to be searched. So especially in big data set, FP-growth is more efficient than APriori.

**b.**

∀X ∈ transaction, buys(X, E)∧buys(X, O) ⇒ buys(X, K) [60%, 100%]

∀X ∈ transaction, buys(X, K)∧ buys(X, O) ⇒ buys(X, E) [60%, 100%]

**6.11**

**Apriori:**

When finding the multiple occurrences of items, we need to treat each item with diﬀerent count value as diﬀerent items, then check if the minimal support is met. For example, B:1 and B:2, that is, B with single count or 2 counts, as diﬀerent items. Then we construct frequent 2-itemsets, 3-itemsets, etc. We need to check the generated itemsets to decide whether they are frequent or not.

**FP Growth:**

When use FP growth method, we also need to consider the frequency of the generated itemset or itemsets. For example, when we do projected DBs, we need to make sure each item is associated with different counts.