732A61/TDDD41 Data Mining - Clustering and Association Analysis

Lecture 7: FP Grow Algorithm

Jose M. Peña
IDA, Linköping University, Sweden

Outline

- Content
  - Frequent Pattern (FP) Grow Algorithm
  - Exercise
  - Summary
- Literature
  - Course book. Second edition: 5.2.4. Third edition: 6.2.4.
  - Han, J., Pei, J. and Yin, Y. Mining Frequent Patterns without Candidate Generation. In Proc. of the 2000 ACM SIGMOD Int. Conf. on Management of Data, 2000.

FP Grow Algorithm

- Assume that we have access to some transactional data, e.g.

Transaction id	Items bought
1	F, A, C, D, G, I, M, P
2	A, B, C, F, L, M, O
3	B, F, H, J, O, W
4	B, C, K, S, P
5	A, F, C, E, L, P, M, N

- The FP grow algorithm returns all the frequent itemsets without candidate generation and, thus, it may save time and space.
- First, it finds frequent 1-itemsets and **sorts the frequent items within each transaction in support descending order**, e.g. with $minsup = 3$

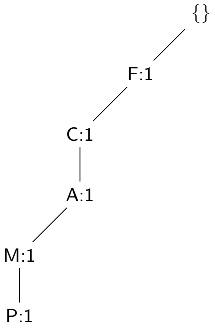
Transaction id	Items bought
1	F, C, A, M, P
2	F, C, A, B, M
3	F, B
4	C, B, P
5	F, C, A, M, P

- Then, it outputs the frequent 1-itemsets, i.e. F, C, A, B, M, and P.

FP Grow Algorithm

- Then, it constructs a so-called FP tree.

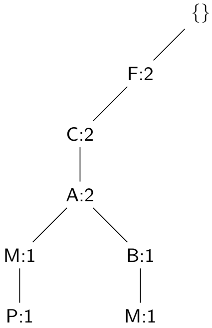
Transaction id	Items bought
1	F, C, A, M, P
2	F, C, A, B, M
3	F, B
4	C, B, P
5	F, C, A, M, P



FP Grow Algorithm

- Then, it constructs a so-called FP tree.

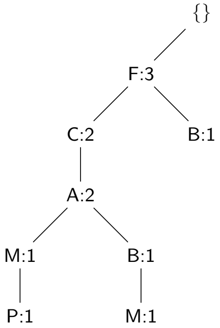
Transaction id	Items bought
1	F, C, A, M, P
2	F, C, A, B, M
3	F, B
4	C, B, P
5	F, C, A, M, P



FP Grow Algorithm

- Then, it constructs a so-called FP tree.

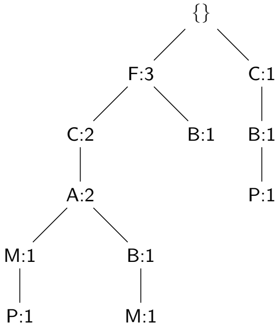
Transaction id	Items bought
1	F, C, A, M, P
2	F, C, A, B, M
3	F, B
4	C, B, P
5	F, C, A, M, P



FP Grow Algorithm

- Then, it constructs a so-called FP tree.

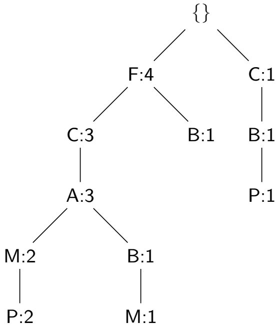
Transaction id	Items bought
1	F, C, A, M, P
2	F, C, A, B, M
3	F, B
4	C, B, P
5	F, C, A, M, P



FP Grow Algorithm

- Then, it constructs a so-called FP tree.

Transaction id	Items bought
1	F, C, A, M, P
2	F, C, A, B, M
3	F, B
4	C, B, P
5	F, C, A, M, P



- Finally, it mines the FP tree for frequent itemsets instead of the original database, since the former is typically much smaller.

FP Grow Algorithm

Algorithm: FP-tree(D, $minsup$)
Input: A transactional database D, and the minimum support $minsup$.
Output: The FP tree for D and $minsup$.

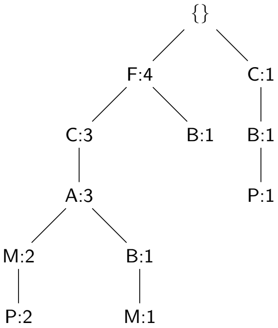
Count support for each item in D
Remove the infrequent items from the transactions in D
Sort the items in each transaction in D in support descending order
Create a FP tree with a single node T with $T.name = NULL$
for each transaction $I \in D$ do
    insert-tree(I, T)

Algorithm: insert-tree($I_1, \ldots I_m$, T)
Input: An itemset $I_1, \ldots, I_m$, and a node T in the FP tree.
Output: Modified FP tree.

if T has a child N such that $N.name = I_1.name$ then
    $N.count++$
else
    create a new child N of T with $N.name = I_1.name$ and $N.count = 1$
if $m > 1$ then
    insert-tree ($I_2, \ldots, I_m, N$)

FP Grow Algorithm

- Given an item X, the X-conditional database consists of all the prefix paths leading to X in the FP tree.



Item	Conditional database
F	-
C	F:3
A	FC:3
B	FCA:1, F:1, C:1
M	FCA:2, FCAB:1
P	FCAM:2, CB:1

- The support of each prefix path in the X-conditional database is equal to the count of X for that prefix path.
- The X-conditional database contains all the itemsets in D that end with X.
- So, it suffices to mine the X-conditional database to find all the frequent itemsets in D that end with X.
- So, re-start the whole process for the X-conditional database, i.e. call the FP grow algorithm recursively.

FP Grow Algorithm

- For instance, the M-conditional database is $\{$FCA:2, FCAB:1$\}$, or

Tid	Items bought
1	F, C, A
2	F, C, A
3	F, C, A, B

- After finding the frequent 1-itemsets and sorting the transactions accordingly, we have

Tid	Items bought
1	F, C, A
2	F, C, A
3	F, C, A

- Output the frequent 1-itemsets, adding M as suffix, i.e. FM, CM, and AM.
- Build the FP tree and the conditional databases.

Item	Conditional database
F	-
C	F:3
A	FC:3

- Re-start the process for the FM-, CM-, and AM-conditional databases.

FP Grow Algorithm

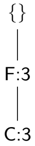
- For instance, the AM-conditional database is $\{$FC:3$\}$, or

Tid	Items bought
1	F, C
2	F, C
3	F, C

- After finding the frequent 1-itemsets and sorting the transactions accordingly, we have

Tid	Items bought
1	F, C
2	F, C
3	F, C

- Output the frequent 1-itemsets, adding AM as suffix, i.e. FAM, and CAM.
- Build the FP tree and the conditional databases.



Item	Conditional database
F	-
C	F:3

- Re-start the process for the FAM-, and CAM-conditional databases.

FP Grow Algorithm

- For instance, the CAM-conditional database is $\{F{:}3\}$, or

Tid	Items bought
1	F
2	F
3	F

- After finding the frequent 1-itemsets and sorting the transactions accordingly, we have

Tid	Items bought
1	F
2	F
3	F

- Output the frequent 1-itemsets, adding CAM as suffix, i.e. FCAM.
- Build the FP tree and the conditional databases.

Item	Conditional database
F	-

- Backtrack.

FP Grow Algorithm

- To mine the FP tree *Tree*, call FP-grow(*Tree*, NULL, *minsup*).

Algorithm: FP-grow(*Tree*, α, *minsup*)
Input: A FP tree *Tree*, an itemset α, and the minimum support *minsup*.
Output: All the itemsets in *Tree* that end with α and have *minsup*.

for each item X in *Tree* do
    output the itemset $\beta = X \cup \alpha$ with support$=X.count$
    build the β conditional database and the corresponding FP tree $Tree_\beta$
    if $Tree_\beta$ is not empty then call FP-grow($Tree_\beta$, β, *minsup*)

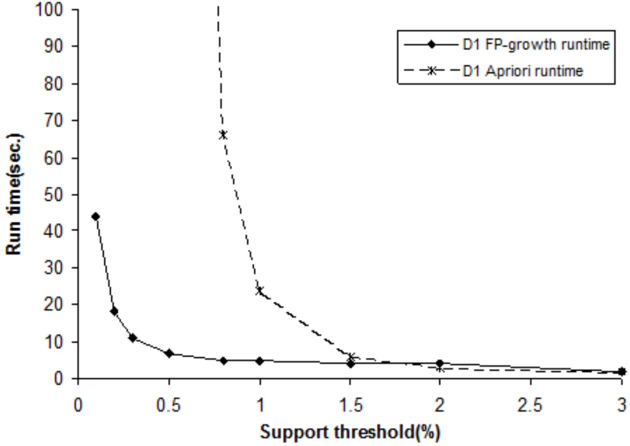
- The algorithm above can be made more efficient by adding the lines below.

if *Tree* has a single branch then
    for each combination β of the nodes in the branch do
        output the itemset $\beta \cup \alpha$ with support $= \min_{X \in \beta} X.count$
else

- The FP grow algorithm is correct, i.e. it misses no frequent itemset.

FP Grow Algorithm

- With small values for *minsup*, there are many and long candidates, which implies long runtime due to expensive operations such as pattern matching, subset checking, storing, etc.



Exercise

- Run the FP grow algorithm on the database below with mininum support equal to two transactions.

Tid	Items bought
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

- Show the execution details (i.e. FP tree construction, conditional databases, recursive calls), not just the frequent itemsets found.

Summary

- Mining transactions to find rules of the form

$$Item_1, \ldots, Item_m \rightarrow Item_{m+1}, \ldots, Item_n$$

  with user-defined minimum support and confidence.
- Two-step solution:
  1. Find all the large itemsets.
  2. Generate all the rules with minimum confidence from the large itemsets.
- We have seen one solution for step 2, and two solutions for step 1, i.e. with and without candidate generation: Apriori and FP grow algorithms.
- Their runtime can differ substantially for small values of *minsup*.