Frequent Itemset Mining

Search for products, brands and more

My Account

More

Cart

Back to top

Frequently bought together

ASUS VivoBook Ultra 14 Core i3 11th Gen - (8 GB/512 GB SSD/Window...

4.5★ (955)

₹43,990 ~~₹56,990~~ 22% off

+

Logitech M90 Wired Optical Mouse

4.4★ (7,198)

₹299

1 Item		1 Add-on		Total
₹43,990	+	₹299	=	₹44,289

ADD 2 ITEMS TO CART

ADD TO CART

BUY NOW

Ratings & Reviews

Rate Product

Roll over image to zoom in

In Stock.

Sold and fulfilled by Cloudtail Pantry.

Item Form	Cream
Brand	Fiama
Special Ingredients	Almond Oil
Scent	Almond
Skin Type	Normal
Package Type	Bottle

See more

About this item

- Moisturised Skin – Contains Skin Conditioners with Moisture Lock that keeps your skin soft
- Soft Skin – Enriched with natural extracts of Brahma kamalam flower, milk cream and Almond oil, it works to keep you fresh and your skin soft

Special offers and product promotions

- **Prime Savings** : 10% Instant Discount up to Rs.1750 with HDFC Bank Debit/Credit Cards (EMI) on Minimum puchase of Rs.5000. Here's how
- **Prime Savings** : Flat Rs.500 Instant Discount with HDFC Bank Debit Cards (Non-EMI) on Minimum puchase of Rs.5000. Here's how
- **Prime Savings** : 10% Instant Discount up to Rs.1250 with HDFC Bank Credit Cards (Non-EMI) on Minimum puchase of Rs.5000. Here's how
- Get GST invoice and save up to 28% on business purchases. Sign up for free Here's how

Frequently bought together

+

+

Total price: ₹328.00

Add all three to Cart

Association Rules Example

Transaction	Items
t_1	Bread, Jelly, Butter
t_2	Bread, Butter
t_3	Bread, Milk, Butter
t_4	Apple, Bread
t_5	Apple, Milk

I = {Apple, Bread, Jelly, Milk, Butter}

Bread => Butter happens pretty frequently

Association rule mining

- Find which item sets are associated
- Association denotes accessing together
- Dataset D is set of transactions T_i
- Each T_i is set of items $I_{ij} \in I$
- Find itemsets A and B such that accessing A implies accessing B

$$A \implies B$$

- Extremely rare that this will happen always
- Not useful if such itemsets occur rarely

Parameters of an Association Rule

- For both A and B to occur, $A \cup B$ must occur
- Two thresholds or parameters
- Support: A and B should occur in at least s (ratio of) transactions

$$P(A, B) = \frac{|A \cup B|}{|T|} \geq s$$

- Confidence: If A occurs, B should occur in at least c (ratio of) transactions

$$P(B|A) = \frac{|A \cup B|}{|A|} \geq c$$

Transaction Id	Itemsets
1	A, C, D
2	B, C, E
3	A, B, C, E
4	B, E

Rule	Support	Confidence
$B \implies E$		
$C \implies E$		
$B, C \implies E$		
$E \implies B$		
$E \implies C$		
$E \implies B, C$		
$A \implies D$		
$D \implies A$		

Mining Association Rules

- Given a support threshold *s* and confidence threshold *c*
  - Mine all frequent itemsets, i.e., support of the itemset is above *s*
  - For association rules from frequent itemsets, i.e., each rule has confidence above *c*

How difficult is the problem?

- What is the naïve approach?
- Generate a candidate itemset
- Test its support
- If frequent, accept
- Else, throw away
- Total number of possible itemsets is $2^n - 1$
- Checking each itemset requires scanning the entire transaction database
- Too impractical

Apriori Principle

- Candidate-generation-and-test paradigm
- Apriori principle: If an itemset is frequent, all its subsets must also be frequent
- Conversely, if an itemset X is infrequent, all its supersets are also infrequent
- This is an *anti-monotonic* property: if a set fails, its supersets fail as well

Apriori Algorithm

- Generates candidate itemsets in order of length
- Tests each such candidate itemset for support threshold
- Uses all frequent itemsets of a particular length to generate candidates having length one more

Transaction Id	Itemsets
0	1, 2, 5
1	2, 4
2	2, 3
3	1, 2, 4
4	1, 3
5	2, 3
6	1, 3
7	1, 2, 3, 5
8	1, 2, 3
9	6

Support threshold $s = 2$

Candidate set C_1

Itemset	Frequency
1	6
2	7
3	6
4	2
5	2
6	1

$\rightarrow$

Frequent set F_1

Itemset	Frequency
1	6
2	7
3	6
4	2
5	2

$\rightarrow$

Candidate set C_2

Itemset	Frequency
1, 2	4
1, 3	4
1, 4	1
1, 5	2
2, 3	4
2, 4	2
2, 5	2
3, 4	0
3, 5	1
4, 5	0

$\rightarrow$

Frequent set F_2

Itemset	Frequency
1, 2	4
1, 3	4
1, 5	2
2, 3	4
2, 4	2
2, 5	2

$\rightarrow$

Candidate set C_3

Itemset	Frequency
1, 2, 3	2
1, 2, 5	2
(1, 3, 5)	subset
(2, 3, 4)	subset
(2, 3, 5)	subset
(2, 4, 5)	subset

$\rightarrow$

Frequent set F_3

Itemset	Frequency
1, 2, 3	2
1, 2, 5	2

Candidate set C_4

Itemset	Frequency
(1, 2, 3, 5)	subset

Do we need to count support of all generated candidates?

- F_3 = {{1, 2, 3}, {1, 2, 4}, {1, 3, 4},
  {1, 3, 5}, {2, 3, 4}}

- After join
  - C_4 = {{1, 2, 3, 4}, {1, 3, 4, 5}, {1,2,3,5}}
- After pruning:
  - C_4 = {{1, 2, 3, 4}}
    because {1, 4, 5} and {1, 2, 5 } are not in F_3 ({1, 3, 4, 5} and {1, 2, 3, 5} are removed)

Ordering Itemsets

- The items in I are sorted in lexicographic order (which is a total order).
- The order is used throughout the algorithm in each itemset.
- $\{w[1], w[2], \ldots, w[k]\}$ represents a k-itemset w consisting of items $w[1], w[2], \ldots, w[k]$, where $w[1] < w[2] < \ldots < w[k]$ according to the total order.

Details: the algorithm

Algorithm Apriori(*T*)

$C_1 \leftarrow$ init-pass(T);

$F_1 \leftarrow \{f \mid f \in C_1, f\text{.count}/n \geq \mathit{minsup}\}$; // n: no. of transactions in T

for (k = 2; $F_{k-1} \neq \emptyset$; k++) **do**

$C_k \leftarrow$ candidate-gen(F_{k-1});

for each transaction $t \in T$ **do**

for each candidate $c \in C_k$ **do**

if c is contained in t **then**

$c.\mathit{count}$++;

end

end

$F_k \leftarrow \{c \in C_k \mid c.\mathit{count}/n \geq \mathit{minsup}\}$

end

return $F \leftarrow \bigcup_k F_k$;

Candidate-gen function

Function candidate-gen(F_{k-1})
    $C_k \leftarrow \emptyset$;
    forall $f_1, f_2 \in F_{k-1}$
            with $f_1 = \{i_1, \ldots, i_{k-2}, i_{k-1}\}$
            and $f_2 = \{i_1, \ldots, i_{k-2}, i'_{k-1}\}$
            and $i_{k-1} < i'_{k-1}$ **do**
        $c \leftarrow \{i_1, \ldots, i_{k-1}, i'_{k-1}\}$; // join f_1 and f_2
        $C_k \leftarrow C_k \cup \{c\}$;
        for each (k-1)-subset s of c **do**
            if ($s \notin F_{k-1}$) **then**
                delete c from C_k; // prune
        end
    end
    return C_k;

Step 2: Generating rules from frequent itemsets

- Frequent itemsets $\neq$ association rules
- One more step is needed to generate association rules
- For each frequent itemset *X*,

  For each proper nonempty subset *A* of *X*,
  - Let *B* = X - *A*
  - A $\rightarrow$ B is an association rule if
    - Confidence(A $\rightarrow$ B) $\geq$ minconf,

      support(A $\rightarrow$ B) = support(A$\cup$B) = support(X)

      confidence(A $\rightarrow$ B) = support(A $\cup$ B) / support(A)

Generating rules: an example

- Suppose {2,3,4} is frequent
- Proper nonempty subsets:
  - {2,3}, {2,4}, {3,4}, {2}, {3}, {4}
- These generate these association rules:
  - 2,3 $\rightarrow$ 4,
  - 2,4 $\rightarrow$ 3,
  - 3,4 $\rightarrow$ 2,
  - 2 $\rightarrow$ 3,4,
  - 3 $\rightarrow$ 2,4,
  - 4 $\rightarrow$ 2,3,

Frequent Pattern Tree

- Frequent pattern (FP)-growth
- Compact representation of entire transaction database as a tree
- FP-tree
- Resembles a prefix tree

Jiawei Han, Jian Pei, and Yiwen Yin. 2000. Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD*. ***[Citations: 9670]***

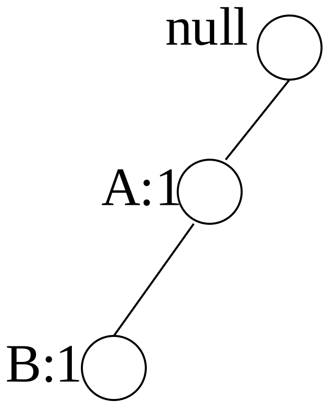
Algorithm

- First finds support of all 1-itemsets
- Items in *descending* order of support forms flist order
- Re-arranges items in every transaction in *flist* order
- Root is “null”
- Nodes are items with corresponding count
- Each transaction is added as a path in the tree
- Count of common prefixes are incremented
- Nodes of same item are linked using node links
- Two database scans

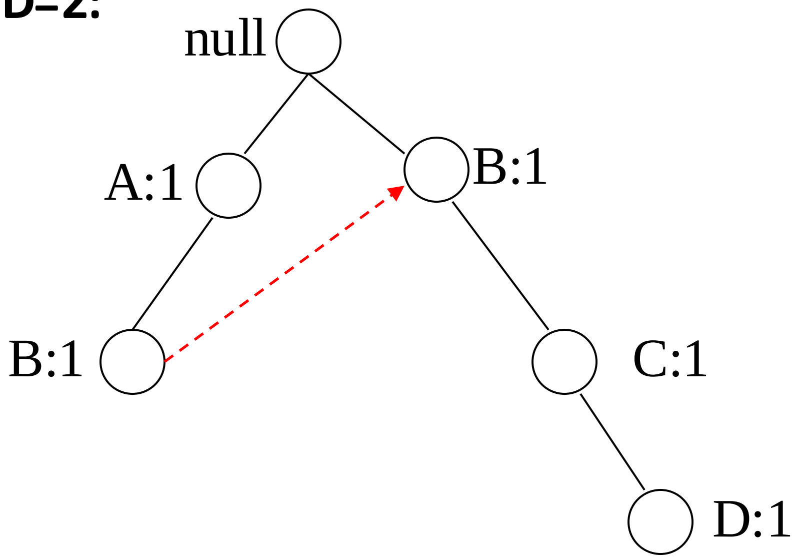
FP-Tree Construction

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

After reading TID=1:



After reading TID=2:



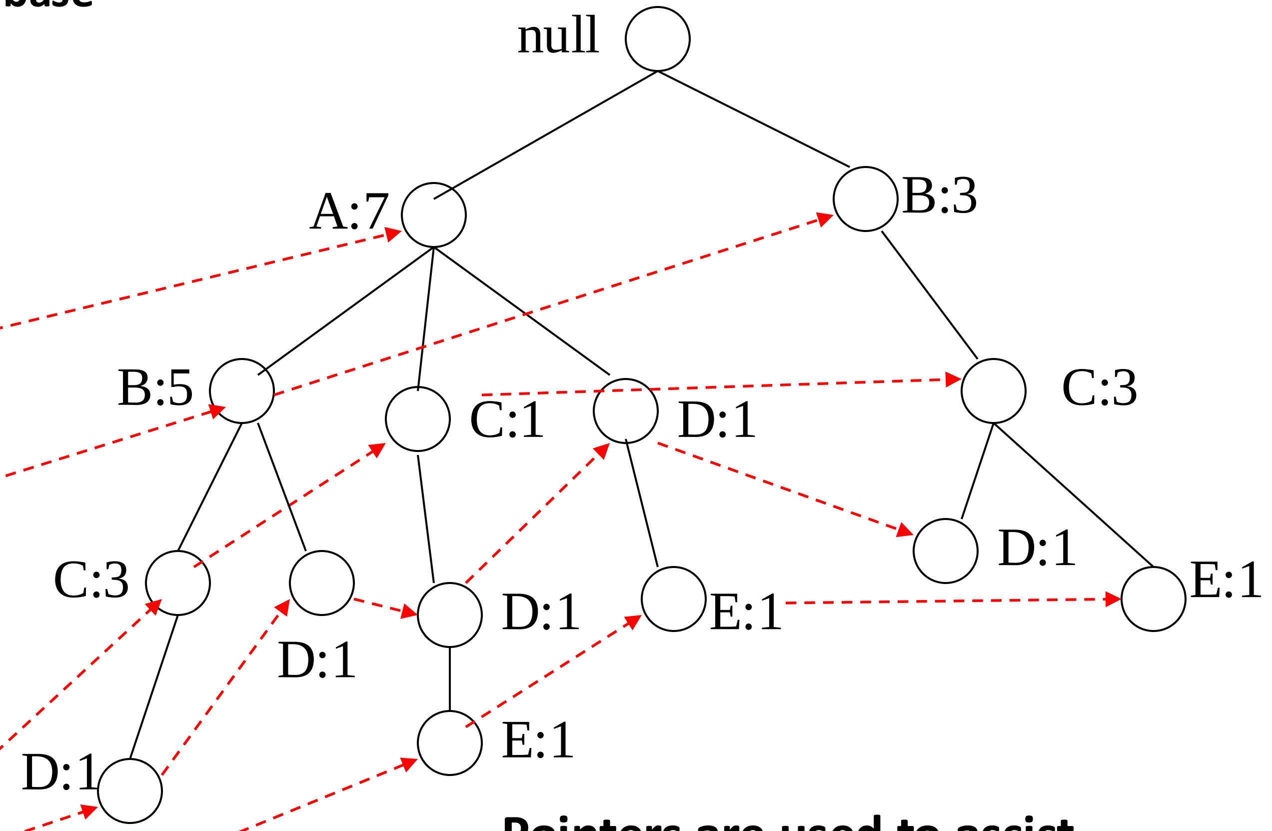
FP-Tree Construction

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

Transaction Database

Header table

Item	Pointer
A	
B	
C	
D	
E	

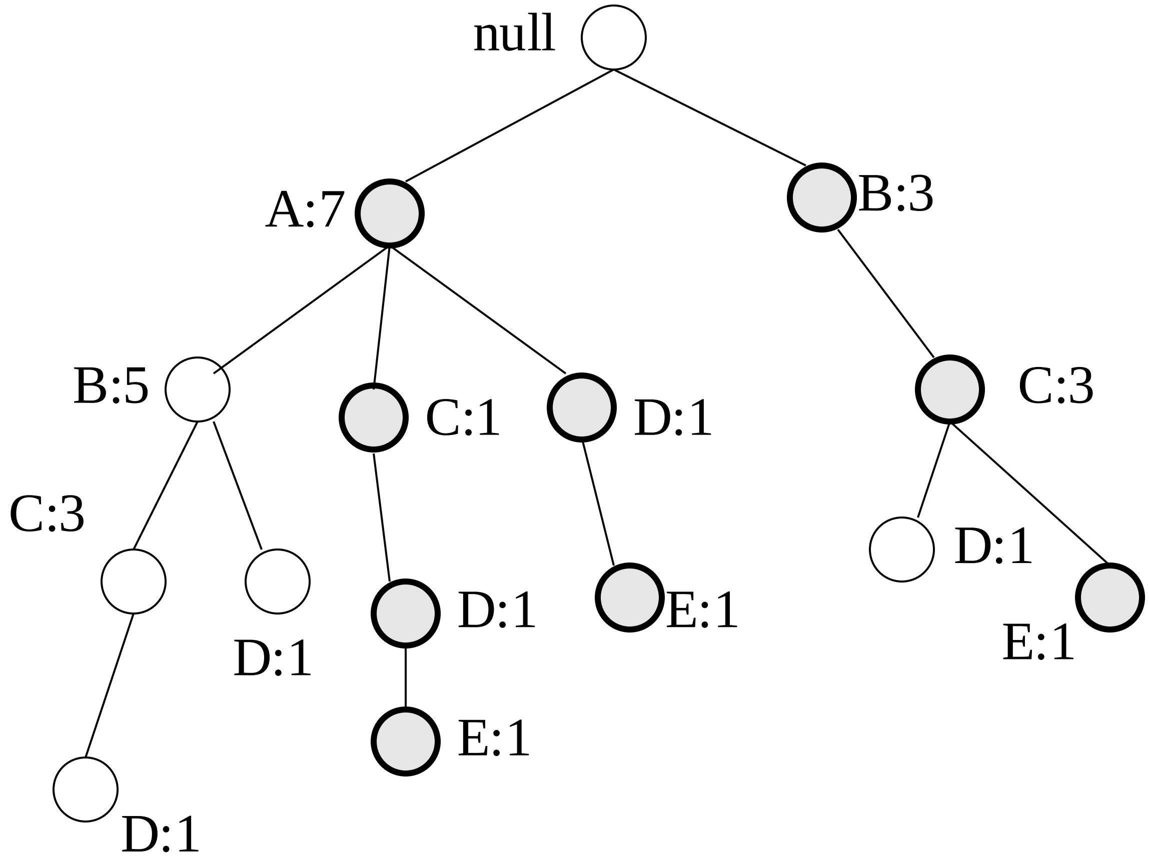


Pointers are used to assist frequent itemset generation

FP-growth

To mine frequent itemsets containing E (starts with the least frequent item)

1. Identify paths containing E
2. Build conditional pattern base



Build conditional pattern base for E:

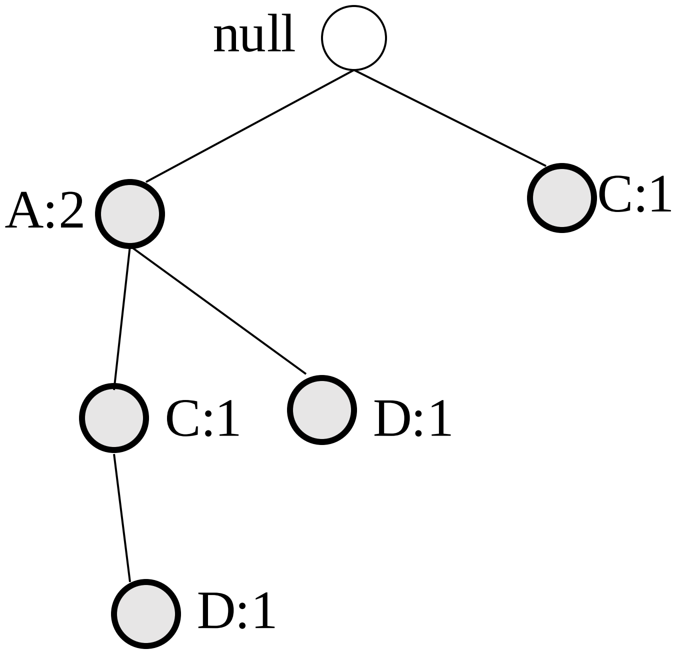
P = {(A:1,C:1,D:1),
(A:1,D:1),
(B:1,C:1)}

Recursively apply FP-growth on P

FP-growth

<u>Conditional tree for E:</u>

Assume minSup=2



Conditional Pattern base for E:

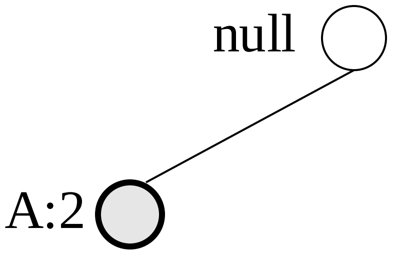
P = {(A:1,C:1,D:1,E:1),
(A:1,D:1,E:1),
(B:1,C:1,E:1)}

Count for E is 3: {E} is frequent itemset

Recursively apply FP-growth on P

FP-growth

<u>Conditional tree for D within conditional tree for E:</u>



Conditional pattern base for D within conditional base for E:

P = {(A:1,C:1,D:1),
(A:1,D:1)}

Count for D is 2: {D,E} is frequent itemset

Recursively apply FP-growth on P

FP-growth

<u>Conditional tree for A within D within E:</u>

Count for A is 2: {A,D,E} is frequent itemset

Next step:

This recursion stops

Construct conditional tree C within conditional tree E

Continue until exploring conditional tree for A (which has only node A)

FP-growth

<u>Conditional tree for C within E:</u>

Assume minSup=2

Conditional Pattern base for CE:

P = {(A:1,C:1,E:1),
(C:1,E:1)}

Output 2:{C,E}

Return since no further recursion

FP-growth

<u>Conditional tree for A within conditional of E:</u>

Assume minSup=2

Conditional Pattern base for E:

P = {(A:2, E:2)}

Output 2:{A,E}

No further recursion.