

# CS685: DATA MINING

## ASSOCIATION RULE MINING

Arnab Bhattacharya  
arnabb@cse.iitk.ac.in

Computer Science and Engineering,  
Indian Institute of Technology, Kanpur  
<http://web.cse.iitk.ac.in/~cs685/>

1<sup>st</sup> semester, 2021-22  
Mon 1030-1200 (online)

# Association Rules

- Find which **itemsets** are associated

# Association Rules

- Find which **itemsets** are associated
- Association denotes accessing together
- Dataset  $D$  is set of transactions  $T_i$
- Each  $T_i$  is set of items  $I_{ij} \in I$

# Association Rules

- Find which **itemsets** 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  $B$

$$A \implies B$$

- This is called an **association rule**

# Association Rules

- Find which **itemsets** 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  $B$

$$A \implies B$$

- This is called an **association rule**
- Called **association rule mining** or **itemset mining** or **basket mining**

# Association Rules

- Find which **itemsets** 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  $B$

$$A \implies B$$

- This is called an **association rule**
- Called **association rule mining** or **itemset mining** or **basket mining**
- Extremely rare that this will happen always

# Association Rules

- Find which **itemsets** 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  $B$

Market Analysis

$$A \implies B$$

$$\{ \text{milk, butter} \} \Rightarrow \{ \text{eggs} \}$$

- This is called an **association rule**
- Called **association rule mining** or **itemset mining** or **basket mining**
- Extremely rare that this will happen always
- Not useful if such itemsets occur rarely

# Parameters of Association Rules

- For both  $A$  and  $B$  to occur,  $A \cup B$  must occur
- Two thresholds or parameters



# Parameters of Association Rules

- 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

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

# Parameters of Association Rules

- 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

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

# Example

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

# Example

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

Rule	Support	Confidence
$B \Rightarrow E$		

# Example

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

Rule	Support	Confidence
$B \Rightarrow E$	0.75	1.00
$C \Rightarrow E$	0.5	2/3

# Example

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

Rule	Support	Confidence
$B \Rightarrow E$	0.75	1.00
$C \Rightarrow E$	0.50	0.67
$B, C \Rightarrow E$		

# Example

$$\begin{aligned}
 &P(E|C) \\
 &= \frac{P(ENC)}{P(C)} \\
 &= \frac{2/4}{3/4} = 2/3
 \end{aligned}$$

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

Rule	Support	Confidence
$B \Rightarrow E$	0.75	1.00
$C \Rightarrow E$	0.50	0.67
$B, C \Rightarrow E$	0.50	1.00
$E \Rightarrow B$	0.75 0.5	1 2/3

# Example

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

Rule	Support	Confidence
$B \Rightarrow E$	0.75	1.00
$C \Rightarrow E$	0.50	0.67
$B, C \Rightarrow E$	0.50	1.00
$E \Rightarrow B$	0.75	1.00
$E \Rightarrow C$		



# Example

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

Rule	Support	Confidence
$B \Rightarrow E$	0.75	1.00
$C \Rightarrow E$	0.50	0.67
$B, C \Rightarrow E$	0.50	1.00
$E \Rightarrow B$	0.75	1.00
$E \Rightarrow C$	0.50	0.67
$E \Rightarrow B, C$		

# Example

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

Rule	Support	Confidence
$B \Rightarrow E$	0.75	1.00
$C \Rightarrow E$	0.50	0.67
$B, C \Rightarrow E$	0.50	1.00
$E \Rightarrow B$	0.75	1.00
$E \Rightarrow C$	0.50	0.67
$E \Rightarrow B, C$	0.50	0.67
$A \Rightarrow D$		

# Example

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

Rule	Support	Confidence
$B \Rightarrow E$	0.75	1.00
$C \Rightarrow E$	0.50	0.67
$B, C \Rightarrow E$	0.50	1.00
$E \Rightarrow B$	0.75	1.00
$E \Rightarrow C$	0.50	0.67
$E \Rightarrow B, C$	0.50	0.67
$A \Rightarrow D$	0.25	0.50
$D \Rightarrow A$		

# Example

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

Rule	Support	Confidence
$B \Rightarrow E$	0.75	1.00
$C \Rightarrow E$	0.50	0.67
$B, C \Rightarrow E$	0.50	1.00
$E \Rightarrow B$	0.75	1.00
$E \Rightarrow C$	0.50	0.67
$E \Rightarrow B, C$	0.50	0.67
$A \Rightarrow D$	0.25	0.50
$D \Rightarrow A$	0.25	1.00

# Definitions

- **k-itemset**: An itemset that contains  $k$  items, i.e., its length is  $k$

# Definitions

- **k-itemset**: An itemset that contains  $k$  items, i.e., its length is  $k$
- **Frequent itemset**: An itemset whose support is more than or equal to the support threshold

# Definitions

- **k-itemset**: An itemset that contains  $k$  items, i.e., its length is  $k$
- **Frequent itemset**: An itemset whose support is more than or equal to the support threshold
- **Infrequent itemset**: An itemset whose support is less than the support threshold

# Definitions

- **k-itemset**: An itemset that contains  $k$  items, i.e., its length is  $k$
- **Frequent itemset**: An itemset whose support is more than or equal to the support threshold
- **Infrequent itemset**: An itemset whose support is less than the support threshold
- **Closed itemset**: An itemset  $X$  for which there does not exist any proper superset  $Y \supset X$  having the same support as  $X$



# Definitions

- **k-itemset**: An itemset that contains  $k$  items, i.e., its length is  $k$
- **Frequent itemset**: An itemset whose support is more than or equal to the support threshold
- **Infrequent itemset**: An itemset whose support is less than the support threshold
- **Closed itemset**: An itemset  $X$  for which there does not exist any proper superset  $Y \supset X$  having the same support as  $X$
- **Maximal frequent itemset** or **Max itemset**: An itemset  $X$  that is frequent and for which there does not exist any proper superset  $Y \supset X$  which is also frequent

# Definitions

- **k-itemset**: An itemset that contains  $k$  items, i.e., its length is  $k$
- **Frequent itemset**: An itemset whose support is more than or equal to the support threshold
- **Infrequent itemset**: An itemset whose support is less than the support threshold
- **Closed itemset**: An itemset  $X$  for which there does not exist any proper superset  $Y \supset X$  having the same support as  $X$
- **Maximal frequent itemset** or **Max itemset**: An itemset  $X$  that is frequent and for which there does not exist any proper superset  $Y \supset X$  which is also frequent
- **Minimal infrequent itemset** or **Min itemset**: An itemset  $X$  that is infrequent and for which there does not exist any proper subset  $Z \subset X$  which is also infrequent

# Definitions

- **k-itemset**: An itemset that contains  $k$  items, i.e., its length is  $k$
- **Frequent itemset**: An itemset whose support is more than or equal to the support threshold
- **Infrequent itemset**: An itemset whose support is less than the support threshold
- **Closed itemset**: An itemset  $X$  for which there does not exist any proper superset  $Y \supset X$  having the same support as  $X$
- **Maximal frequent itemset** or **Max itemset**: An itemset  $X$  that is frequent and for which there does not exist any proper superset  $Y \supset X$  which is also frequent
- **Minimal infrequent itemset** or **Min itemset**: An itemset  $X$  that is infrequent and for which there does not exist any proper subset  $Z \subset X$  which is also infrequent
- **Strong rule**: An association rule whose confidence is more than or equal to the confidence threshold

# Definitions

- **k-itemset**: An itemset that contains  $k$  items, i.e., its length is  $k$
- **Frequent itemset**: An itemset whose support is more than or equal to the support threshold
- **Infrequent itemset**: An itemset whose support is less than the support threshold
- **Closed itemset**: An itemset  $X$  for which there does not exist any proper superset  $Y \supset X$  having the same support as  $X$
- **Maximal frequent itemset** or **Max itemset**: An itemset  $X$  that is frequent and for which there does not exist any proper superset  $Y \supset X$  which is also frequent
- **Minimal infrequent itemset** or **Min itemset**: An itemset  $X$  that is infrequent and for which there does not exist any proper subset  $Z \subset X$  which is also infrequent
- **Strong rule**: An association rule whose confidence is more than or equal to the confidence threshold
- **Weak rule**: An association rule whose confidence is less than the confidence threshold

# Finding Association Rules

- Mining association rules require two steps
  - Finding **frequent** itemsets
  - Generating **strong** association rules

# Finding Association Rules

- Mining association rules require two steps
  - Finding **frequent** itemsets
  - Generating **strong** association rules
- The first step is more time-consuming

# Brute-force Algorithm

- Generate a candidate itemset
- Test its support
- If frequent, accept
- Else, throw away

# Brute-force Algorithm

- 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

# 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
- Stop till there is no more candidate or when length is exhausted

# 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
- Stop till there is no more candidate or when length is exhausted
- Candidate itemsets of length  $k$  is  $C_k$
- Frequent itemsets of length  $k - 1$  is  $F_{k-1}$
- **Join step:**  $C_k = F_{k-1} \bowtie F_{k-1}$ 
  - Join two candidates whose  $k - 2$  items are common
  - Perform **subset checking**
- **Prune step:**  $F_k = \{I \in C_k : |I| \geq s\}$ 
  - Retain only frequent itemsets

# 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
- Stop till there is no more candidate or when length is exhausted
- Candidate itemsets of length  $k$  is  $C_k$
- Frequent itemsets of length  $k - 1$  is  $F_{k-1}$
- **Join step:**  $C_k = F_{k-1} \bowtie F_{k-1}$ 
  - Join two candidates whose  $k - 2$  items are common
  - Perform **subset checking**
- **Prune step:**  $F_k = \{I \in C_k : |I| \geq s\}$ 
  - Retain only frequent itemsets
- Requires  $k$  database scans for itemsets up to length  $k$

# Apriori Example

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  $\underline{s = 2}$

# Apriori example (contd.)

Candidate set  $C_1$

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

→

# Apriori example (contd.)

Candidate set  $C_1$

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

→

Frequent set  $F_1$

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

→



# Apriori example (contd.)

Candidate set  $C_1$

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

→

Frequent set  $F_1$

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

→

Candidate set  $C_2$

Itemset	Frequency
1, 2	4
1, 3	4
<del>1, 4</del>	<del>1</del>
1, 5	2
2, 3	4
2, 4	2
2, 5	2
<del>3, 4</del>	<del>0</del>
<del>3, 5</del>	<del>1</del>
<del>4, 5</del>	<del>0</del>

→

# Apriori example (contd.)

Candidate set  $C_1$

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

→

Frequent set  $F_1$

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

→

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

→

Frequent set  $F_2$

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

→

# Apriori example (contd.)

Candidate set  $C_1$

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

→

Frequent set  $F_1$

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

→

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

→

Frequent set  $F_2$

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

→

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

→

# Apriori example (contd.)

Candidate set  $C_1$

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

→

Frequent set  $F_1$

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

→

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

→

Frequent set  $F_2$

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

→

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

→

Frequent set  $F_3$

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

# Apriori example (contd.)

Candidate set  $C_1$

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

→

Frequent set  $F_1$

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

→

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

→

Frequent set  $F_2$

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

→

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

→

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

# Partitioning

- *Transaction-wise partitioning*

- Partition transactions into different sets
- Find frequent and infrequent itemsets in each partition with support threshold  $s'$  (according to ratio of transactions in each partition)
  - For two equal partitions,  $s' = s/2$

- ✓ • Report all itemsets that are frequent in all partitions
- ✓ • Prune all itemsets that are infrequent in all partitions

frequent in all  $\Rightarrow$  overall frequent  
infrequent in all  $\Rightarrow$  overall infrequent

# Partitioning

- *Transaction-wise partitioning*

- Partition transactions into different sets
- Find frequent and infrequent itemsets in each partition with support threshold  $s'$  (according to ratio of transactions in each partition)
  - For two equal partitions,  $s' = s/2$
- Report all itemsets that are frequent in all partitions
- Prune all itemsets that are infrequent in all partitions

- *Item-wise partitioning*

- Partition items into different sets
- Find frequent itemsets in each partition
- Join only these frequent itemsets to form global candidates

*Join*

# FP-Growth

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



# FP-Growth



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

# FP-Growth

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

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

# FP-Growth

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

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  $s = 2$

# FP-Tree Example

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

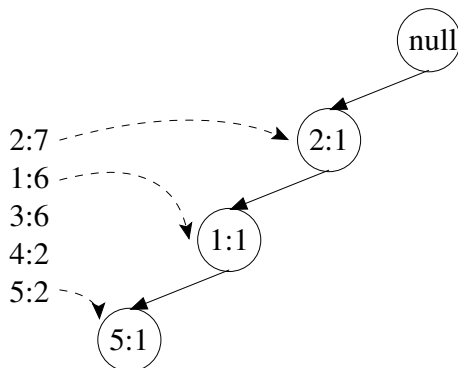
Support threshold  $s = 2$

→

Flist order of items	
Item	Frequency
2	7
1	6
3	6
4	2
5	2

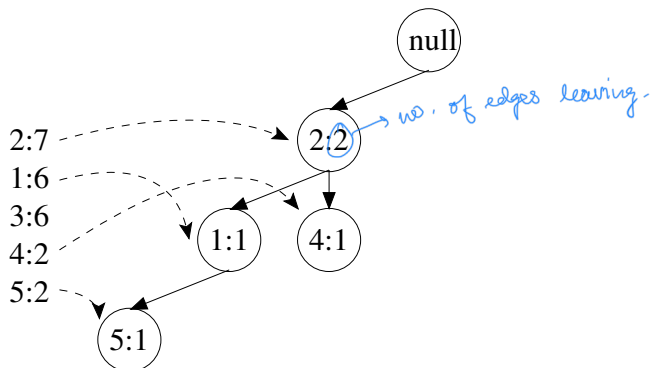
# FP-Tree Construction

- Adding transaction 0: 2, 1, 5



# FP-Tree Construction (contd.)

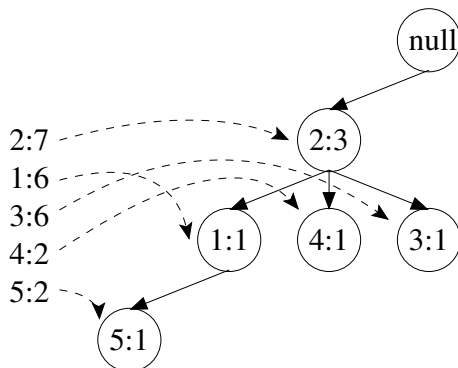
- Adding transaction 1: 2, 4





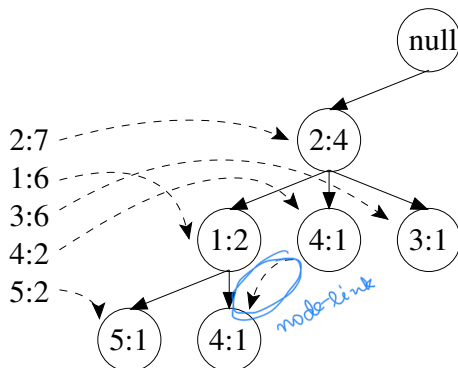
# FP-Tree Construction (contd.)

- Adding transaction 2: 2, 3



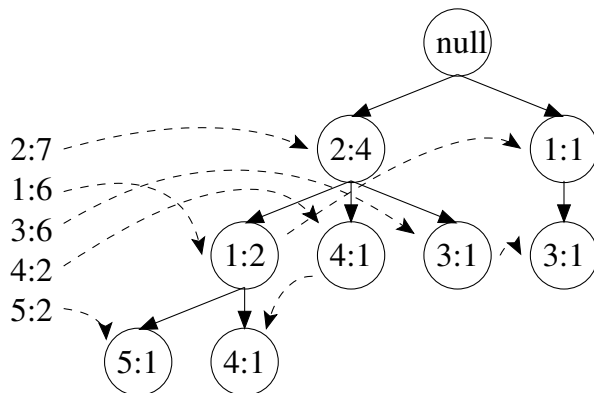
# FP-Tree Construction (contd.)

- Adding transaction 3: 2, 1, 4



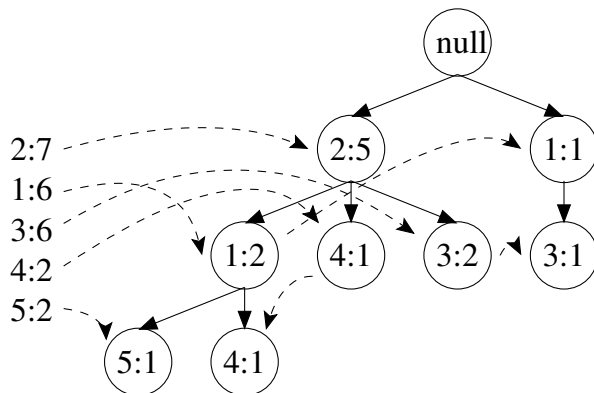
# FP-Tree Construction (contd.)

- Adding transaction 4: 1, 3



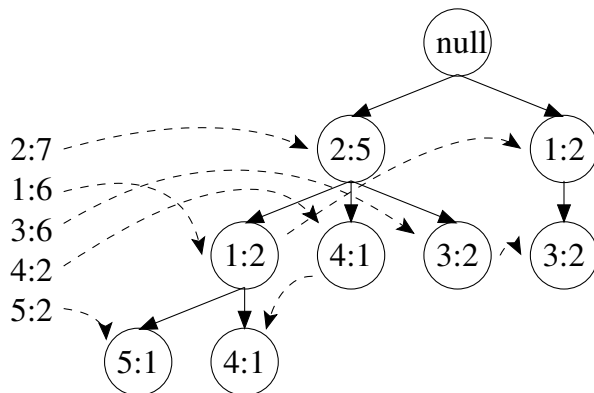
# FP-Tree Construction (contd.)

- Adding transaction 5: 2, 3



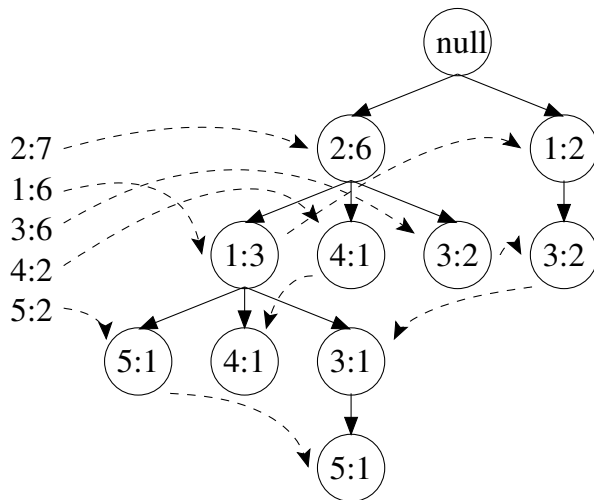
# FP-Tree Construction (contd.)

- Adding transaction 6: 1, 3



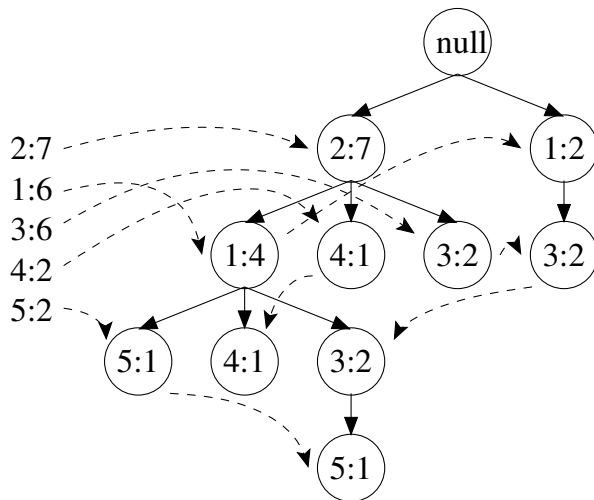
# FP-Tree Construction (contd.)

- Adding transaction 7: 2, 1, 3, 5



# FP-Tree Construction (contd.)

- Adding transaction 8: 2, 1, 3



# FP-Tree Mining

- Starts with the item with the least support, say  $x$
- *Projects* its paths from the base tree
- $x$  is the suffix in all such paths



# FP-Tree Mining

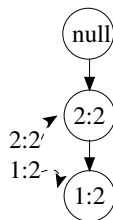
- Starts with the item with the least support, say  $x$
- *Projects* its paths from the base tree
- $x$  is the suffix in all such paths
- A new FP-tree is built with only these paths (equivalently, transactions) with  $x$  removed
- This new FP-tree is *recursively* mined to find frequent patterns
- All such frequent patterns are appended with  $x$  and returned

# FP-Tree Mining

- Starts with the item with the least support, say  $x$
- *Projects* its paths from the base tree
- $x$  is the suffix in all such paths
- A new FP-tree is built with only these paths (equivalently, transactions) with  $x$  removed
- This new FP-tree is *recursively* mined to find frequent patterns
- All such frequent patterns are appended with  $x$  and returned
- The item with the next lowest count is continued with

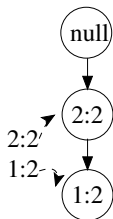
# FP-Tree Mining Example

- For the least frequent item: 5
- Two prefix paths found by traversing node links are (2, 1): 1 and (2, 1, 3): 1
- This forms the **conditional pattern base**
- 3 is discarded as its support ( $= 1$ ) is less than threshold
- From conditional pattern base, **conditional FP-tree** is then constructed



# FP-Tree Mining Example

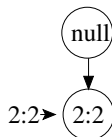
- For the least frequent item: 5
- Two prefix paths found by traversing node links are (2, 1): 1 and (2, 1, 3): 1
- This forms the **conditional pattern base**
- 3 is discarded as its support (= 1) is less than threshold
- From conditional pattern base, **conditional FP-tree** is then constructed



- Frequent patterns found are (1, 5): 2, (2, 1, 5): 2 and (2, 5): 2

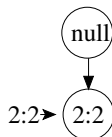
# FP-Tree Mining Example (contd.)

- For the next least frequent item: 4
- Two prefix paths found by traversing node links are (2, 1): 1 and (2): 1
- This forms the **conditional pattern base**
- 1 is discarded as its support ( $= 1$ ) is less than threshold
- From conditional pattern base, **conditional FP-tree** is then constructed



## FP-Tree Mining Example (contd.)

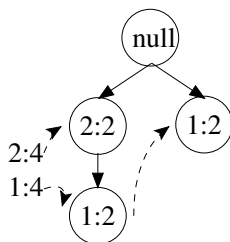
- For the next least frequent item: 4
- Two prefix paths found by traversing node links are (2, 1): 1 and (2): 1
- This forms the **conditional pattern base**
- 1 is discarded as its support ( $= 1$ ) is less than threshold
- From conditional pattern base, **conditional FP-tree** is then constructed



- Frequent patterns found are (2, 4): 2

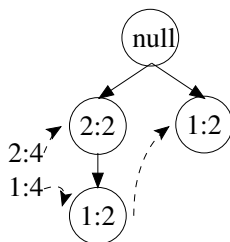
# FP-Tree Mining Example (contd.)

- For the next least frequent item: 3
- Three prefix paths found by traversing node links are (2, 1): 2, (2): 2 and (1): 2
- This forms the **conditional pattern base**
- From conditional pattern base, **conditional FP-tree** is then constructed



# FP-Tree Mining Example (contd.)

- For the next least frequent item: 3
- Three prefix paths found by traversing node links are (2, 1): 2, (2): 2 and (1): 2
- This forms the **conditional pattern base**
- From conditional pattern base, **conditional FP-tree** is then constructed

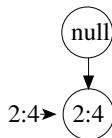


- Frequent patterns found are (1, 3): 4, (2, 1, 3): 2 and (2, 3): 4



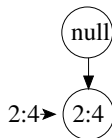
# FP-Tree Mining Example (contd.)

- For the next least frequent item: 1
- One prefix path found by traversing node links is (2, 1): 4
- This forms the **conditional pattern base**
- From conditional pattern base, **conditional FP-tree** is then constructed



# FP-Tree Mining Example (contd.)

- For the next least frequent item: 1
- One prefix path found by traversing node links is (2, 1): 4
- This forms the **conditional pattern base**
- From conditional pattern base, **conditional FP-tree** is then constructed



- Frequent patterns found are (2, 1): 4

## FP-Tree Mining Example (contd.)

- For the most frequent item: 2
- Nothing needs to be done
- Assumption is that all 1-itemsets are already returned

# Projected and Residual Databases

- Consider the item with the largest support, say  $x$
- $x$  partitions transactions into two parts
- Transactions containing  $x$  form the **projected database**  $P_x$  of  $x$
- Transactions after removing  $x$  form the **residual database**  $R_x$  of  $x$

# Projected and Residual Databases

- Consider the item with the largest support, say  $x$
- $x$  partitions transactions into two parts
- Transactions containing  $x$  form the **projected database**  $P_x$  of  $x$
- Transactions after removing  $x$  form the **residual database**  $R_x$  of  $x$
- Each partition is mined recursively by considering the next frequent item, say  $y$

# Projected and Residual Databases

- Consider the item with the largest support, say  $x$
- $x$  partitions transactions into two parts
- Transactions containing  $x$  form the **projected database**  $P_x$  of  $x$
- Transactions after removing  $x$  form the **residual database**  $R_x$  of  $x$
- Each partition is mined recursively by considering the next frequent item, say  $y$
- Union of frequent itemsets from both partitions produce the final set of frequent itemsets

# Projected and Residual Databases

- Consider the item with the largest support, say  $x$
- $x$  partitions transactions into two parts
- Transactions containing  $x$  form the **projected database**  $P_x$  of  $x$
- Transactions after removing  $x$  form the **residual database**  $R_x$  of  $x$
- Each partition is mined recursively by considering the next frequent item, say  $y$
- Union of frequent itemsets from both partitions produce the final set of frequent itemsets
- Why item with largest support?
  - Any item works
  - Item with largest support reduces work

# Projected and Residual Databases

- Consider the item with the largest support, say  $x$
- $x$  partitions transactions into two parts
- Transactions containing  $x$  form the **projected database**  $P_x$  of  $x$
- Transactions after removing  $x$  form the **residual database**  $R_x$  of  $x$
- Each partition is mined recursively by considering the next frequent item, say  $y$
- Union of frequent itemsets from both partitions produce the final set of frequent itemsets
- Why item with largest support?
  - Any item works
  - Item with largest support reduces work
- Consider any (in-)frequent itemset  $I$ 
  - If  $x \in I$ , then it will be (in-)frequent in  $P_x$  as well
  - If  $x \notin I$ , then it will be (in-)frequent in  $R_x$  as well
    - Frequency of  $I$  does not change in  $R_x$



- **H-mine** is a partitioning-based algorithm
- It first sorts the items in *flist* order
- From each item, a pointer is linked to the first transaction that contain this item as the first in flist order
- All subsequent transactions of the same nature are chained
- Following the chain produces the projected database for that item
- The frequent itemsets are mined recursively then

# Mining Closed and Maximally Frequent Itemsets

- Apriori algorithm works
- When checking candidates, check subsets
- If any subset has same support, remove that subset

# Mining Closed and Maximally Frequent Itemsets

- Apriori algorithm works
- When checking candidates, check subsets
- If any subset has same support, remove that subset
- Apriori may be run in reverse direction, starting with all items and then generating subsets as candidates

# Mining Closed and Maximally Frequent Itemsets

- Apriori algorithm works
- When checking candidates, check subsets
- If any subset has same support, remove that subset
- Apriori may be run in reverse direction, starting with all items and then generating subsets as candidates
- A single support threshold across all itemset lengths may not be useful
- Chances of itemsets with larger length occurring are less
- **MLMS** model: **Multiple Length Minimum Support**
- Apriori works again
- If support at lesser length is smaller, e.g.,  $s_k < s_{k+1}$ 
  - All  $k$ -length subsets of frequent itemsets of length  $k + 1$  are frequent
  - Conversely, if an itemset is pruned at length  $k$ , all its supersets of length  $k + 1$  will be infrequent

# Are Strong Association Rules Always Good?

- Consider the original transaction database
- Consider the rule  $3 \implies 2$
- Support is 0.4 and confidence is 0.67

# Are Strong Association Rules Always Good?

- Consider the original transaction database
- Consider the rule  $3 \implies 2$
- Support is 0.4 and confidence is 0.67
- However, support of 2 itself is 0.7

# Are Strong Association Rules Always Good?

- Consider the original transaction database
- Consider the rule  $3 \implies 2$
- Support is 0.4 and confidence is 0.67
- However, support of 2 itself is 0.7
- When there is no influence, 2 occurs more frequently than when 3 is there
- The effect of 3 is thus *negative* on 2
- Just support and confidence thresholds are, therefore, not enough

# Lift

- A correlation measure **lift**
- Lift measures how correlated the two itemsets are

$$\text{lift}(A \rightarrow B) = \text{confidence}(A \rightarrow B) / \text{support}(B)$$



- A correlation measure **lift**
- Lift measures how correlated the two itemsets are

$$\text{lift}(A \rightarrow B) = \text{confidence}(A \rightarrow B) / \text{support}(B)$$

- In terms of probabilities

$$\begin{aligned}\text{lift}(A \rightarrow B) &= \frac{P(A \cup B) / P(A)}{P(B)} \\ &= \frac{P(A \cup B)}{P(A) \cdot P(B)}\end{aligned}$$

- Lift is *symmetric*
- If lift is 1,  $A$  and  $B$  are *independent*
- If lift is  $< 1$ , they are *negatively* correlated
- If lift is  $> 1$ , they are *positively* correlated

- A correlation measure **lift**
- Lift measures how correlated the two itemsets are

$$\text{lift}(A \rightarrow B) = \text{confidence}(A \rightarrow B) / \text{support}(B)$$

- In terms of probabilities

$$\begin{aligned}\text{lift}(A \rightarrow B) &= \frac{P(A \cup B) / P(A)}{P(B)} \\ &= \frac{P(A \cup B)}{P(A) \cdot P(B)}\end{aligned}$$

- Lift is *symmetric*
- If lift is 1,  $A$  and  $B$  are *independent*
- If lift is  $< 1$ , they are *negatively* correlated
- If lift is  $> 1$ , they are *positively* correlated
- Lift of the rule  $3 \implies 2$  is  $0.67/0.7 = 0.95$
- Thus, 3 and 2 are negatively correlated

# Probabilistic Association Rule Mining

- Occurrence of an item in a transaction is not just presence or absence
- It is present with a probability  $p \in [0, 1]$
- Applications
  - Medical: a patient may have cancer with 70% chance, hepatitis with 10% chance, etc.

Transaction id	Item A	Item B	Item C	Item D
0	0.9	0.8	0.0	0.2
1	0.7	0.7	1.0	0.3
2	0.2	0.5	0.9	0.5

- Support of 1-itemsets can be found by just adding the columns
- Support of larger itemsets can be found by adding the products of the corresponding probabilities
  - Support of (A) is  $0.9 + 0.7 + 0.2 = 1.8$
  - Support of (A,B) is  $0.9 \times 0.8 + 0.7 \times 0.7 + 0.2 \times 0.5 = 1.31$

# Probabilistic Association Rule Mining

- Occurrence of an item in a transaction is not just presence or absence
- It is present with a probability  $p \in [0, 1]$
- Applications
  - Medical: a patient may have cancer with 70% chance, hepatitis with 10% chance, etc.

Transaction id	Item A	Item B	Item C	Item D
0	0.9	0.8	0.0	0.2
1	0.7	0.7	1.0	0.3
2	0.2	0.5	0.9	0.5

- Support of 1-itemsets can be found by just adding the columns
- Support of larger itemsets can be found by adding the products of the corresponding probabilities
  - Support of (A) is  $0.9 + 0.7 + 0.2 = 1.8$
  - Support of (A,B) is  $0.9 \times 0.8 + 0.7 \times 0.7 + 0.2 \times 0.5 = 1.31$
- More general model

# Probabilistic Association Rule Mining

- Occurrence of an item in a transaction is not just presence or absence
- It is present with a probability  $p \in [0, 1]$
- Applications
  - Medical: a patient may have cancer with 70% chance, hepatitis with 10% chance, etc.

Transaction id	Item A	Item B	Item C	Item D
0	0.9	0.8	0.0	0.2
1	0.7	0.7	1.0	0.3
2	0.2	0.5	0.9	0.5

- Support of 1-itemsets can be found by just adding the columns
- Support of larger itemsets can be found by adding the products of the corresponding probabilities
  - Support of (A) is  $0.9 + 0.7 + 0.2 = 1.8$
  - Support of (A,B) is  $0.9 \times 0.8 + 0.7 \times 0.7 + 0.2 \times 0.5 = 1.31$
- More general model
- Apriori can be modified easily to work

# Probabilistic Association Rule Mining

- Occurrence of an item in a transaction is not just presence or absence
- It is present with a probability  $p \in [0, 1]$
- Applications
  - Medical: a patient may have cancer with 70% chance, hepatitis with 10% chance, etc.

Transaction id	Item A	Item B	Item C	Item D
0	0.9	0.8	0.0	0.2
1	0.7	0.7	1.0	0.3
2	0.2	0.5	0.9	0.5

- Support of 1-itemsets can be found by just adding the columns
- Support of larger itemsets can be found by adding the products of the corresponding probabilities
  - Support of (A) is  $0.9 + 0.7 + 0.2 = 1.8$
  - Support of (A,B) is  $0.9 \times 0.8 + 0.7 \times 0.7 + 0.2 \times 0.5 = 1.31$
- More general model
- Apriori can be modified easily to work
- FP-tree and H-mine cannot be ✓