

# **723A75 Advanced Data Mining**

## **TDDD41 Data Mining - Clustering and Association Analysis**

### Lecture 8: Constrained Frequent Itemset Mining

---

Johan Alenlöv  
IDA, Linköping University, Sweden

- Content
  - Recap
  - Monotone and antimonotone constraints
  - Apriori algorithm with constraints
  - FP grow algorithm with constraints
  - Convertible (anti) monotone constraints
  - Summary
- Literature
  - Course Book. 2nd ed.: 5.5. 3rd ed.: 7.3
  - Pei, J., and Han, J. **Can We Push More Constraints into Frequent Pattern Mining?**. In Proc. of the 2000 Int. Conf. on Knowledge Discovery and Data Mining, 2000.

## Recap: 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  $minsup$ .

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

**Algorithm:** insert-tree( $I_1, \dots, I_m, T$ )

**Input:** An itemset  $I_1, \dots, I_m$ , and a node  $T$  in the FP tree.

**Output:** Modified FP tree.

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

## Recap: FP Grow Algorithm

- To mine the FP tree  $Tree$ , call  $FP\text{-}grow(Tree, \text{NULL}, \text{minsup})$ .

**Algorithm:**  $FP\text{-}grow(Tree, \alpha, \text{minsup})$

**Input:** A FP tree  $Tree$ , an itemset  $\alpha$ , and the minimum support  $\text{minsup}$ .

**Output:** All the itemsets in  $Tree$  that end with  $\alpha$  and have  $\text{minsup}$ .

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

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

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

- The FP grow algorithm is correct.

- Run the FP grow algorithm on the database below with minsup 2.

Tid	Items
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.

# Monotone and Antimonotone Constraints

- A constraint is a function that returns **true** or **false** for every itemset.
- It tells us if the itemset satisfies the constraint or not.
  - The itemset has support equal or greater than a given value.
  - The sum of the prices of the items in the itemset is greater than a given value.
  - The most expensive item in the itemset cost less than a given value.
  - The itemset contains a specific value.
  - The itemset contains exactly a certain number of items.
- A constraint  $C$  is monotone when for every itemset  $A$  and  $B$  such that  $A \subseteq B$ , if  $C(A) = \text{true}$  then  $C(B) = \text{true}$ .
  - The itemset contains a certain item.
- A constraint  $C$  is antimonotone when for every itemset  $A$  and  $B$  such that  $A \subseteq B$ , if  $C(B) = \text{true}$  then  $C(A) = \text{true}$ .
  - The support of the itemset is equal or greater than a given value.
- Alternatively,  $C$  is antimonotone when for every itemset  $A$  and  $B$  such that  $A \subseteq B$ , if  $C(A) = \text{false}$  then  $C(B) = \text{false}$ .
- Note that the **apriori property** applies to every antimonotone constrain, i.e. no need to check the constraint for supersets of  $A$  if  $C(A) = \text{false}$ .

## Examples of Monotone and Antimonotone Constraints

- Here  $S$  is the set of prices in the itemset and  $v$  is a given values.
- Examples of monotone constraints:
  - $\text{sum}(S) \geq v$
  - $\text{min}(S) \leq v$
  - $\text{range}(S) \geq v$
- Examples of antimonotone constraints:
  - $\text{sum}(S) \leq v$
  - $\text{max}(S) \leq v$
  - $\text{range}(S) \leq v$

## Examples of Monotone and Antimonotone Constraints

Constraint	Antimonotone	Monotone
$v \in S$	No	Yes
$S \supseteq V$	No	Yes
$S \subseteq V$	Yes	No
$\min(S) \leq v$	No	Yes
$\min(S) \geq v$	Yes	No
$\max(S) \leq v$	Yes	No
$\max(S) \geq v$	No	Yes
$\text{count}(S) \leq v$	Yes	No
$\text{count}(S) \geq v$	No	Yes
$\text{sum}(S) \leq v \ (a \in S, a \geq 0)$	Yes	No
$\text{sum}(S) \geq v \ (a \in S, a \geq 0)$	No	Yes
$\text{range}(S) \leq v$	Yes	No
$\text{range}(S) \geq v$	No	Yes
$\text{avg}(S) \theta v, \theta \in \{\leq, \geq\}$	No but convertible	No but convertible
$\text{support}(S) \geq \xi$	Yes	No
$\text{support}(S) \leq \xi$	No	Yes



**Algorithm:** apriori( $D$ ,  $minsup$ ,  $C$ )

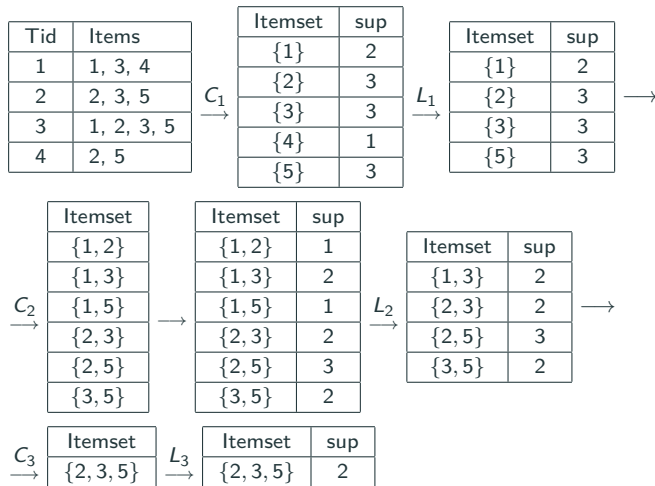
**Input:** A transactional database  $D$ ,  $minsup$ , and an antimonotone constraint  $C$ .

**Output:** All the large itemsets in  $D$  that satisfy  $C$ .

```
1   $L_1 = \{ \text{large 1-itemsets that satisfy } C \}$ 
2  for ( $k = 2; L_{k-1} \neq \emptyset; k++$ ) do
3       $C_k = \text{apriori-gen}(L_{k-1})$     // Generate candidate large  $k$ -itemsets
4      for all  $t \in D$  do
5          for all  $c \in C_k$  such that  $c \in t$  do
6               $c.count++$ 
7       $L_k = \{c \in C_k \mid c.count \geq minsup \text{ and } C(c)=true\}$ 
8  return  $\bigcup_k L_k$ 
```

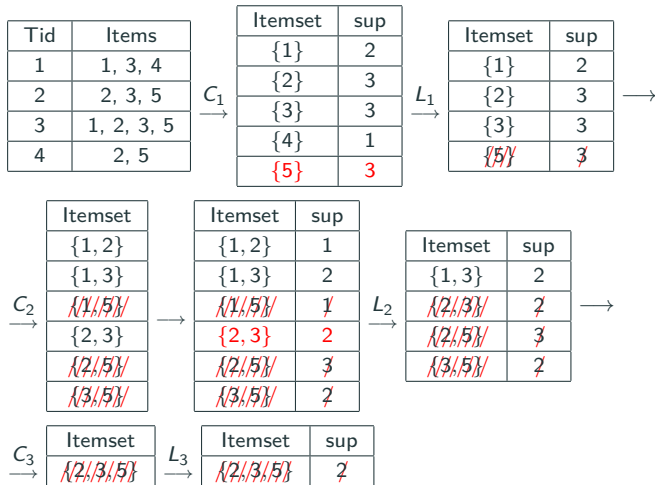
# Apriori Algorithm with Antimonotone Constraint

Run the Apriori algorithm with the following database with minsup 2 and constraint  $\text{sum}(S) < 5$  where the item price coincides with the item id.



# Apriori Algorithm with Antimonotone Constraint

Run the Apriori algorithm with the following database with minsup 2 and constraint  $\text{sum}(S) < 5$  where the item price coincides with the item id.



**Algorithm:** apriori( $D$ ,  $minsup$ ,  $C$ )

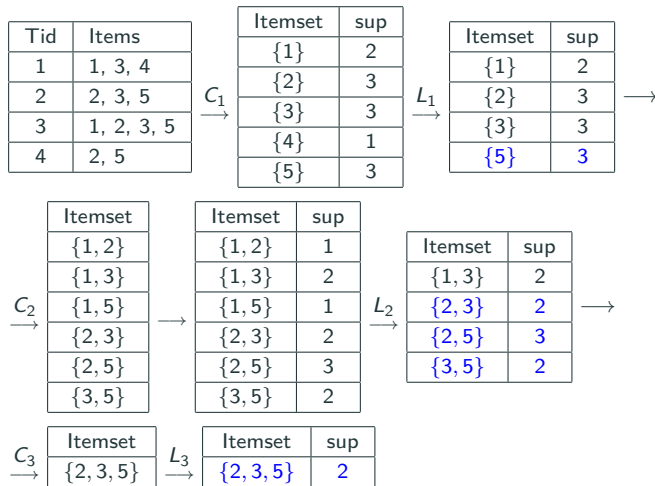
**Input:** A transactional database  $D$ ,  $minsup$ , and a monotone constraint  $C$ .

**Output:** All the large itemsets in  $D$  that satisfy  $C$ .

```
1   $L_1 = \{ \text{large 1-itemsets} \}$ 
2  for ( $k = 2; L_{k-1} \neq \emptyset; k++$ ) do
3       $C_k = \text{apriori-gen}(L_{k-1})$  // Generate candidate large  $k$ -itemsets
4      for all  $t \in D$  do
5          for all  $c \in C_k$  such that  $c \in t$  do
6               $c.count++$ 
7       $L_k = \{c \in C_k \mid c.count \geq minsup\}$ 
8  return  $\{c \in \bigcup_k L_k \mid C(c) = \text{true or } C(d) = \text{true for some } d \subseteq c\}$ 
```

# Apriori Algorithm with Monotone Constraint

Run the Apriori algorithm with the following database with minsup 2 and constraint  $\text{sum}(S) \geq 5$  where the item price coincides with the item id.



**Algorithm:** FP-grow( $Tree, \alpha, minsup, C$ )

**Input:** A FP tree  $Tree$ , an itemset  $\alpha$ ,  $minsup$ , and a monotone constraint  $C$ .

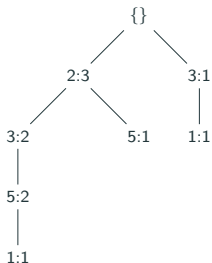
**Output:** All the itemsets in  $Tree$  that end with  $\alpha$ , have  $minsup$  and satisfy  $C$ .

- 1 **if**  $C(\alpha) = true$  **then**
- 2     **replace**  $C$  **with**  $C_{true}$      //  $C_{true}$  is a constraint that always returns true
- 3     for each item  $X$  in  $Tree$  do
- 4         output the itemset  $\beta = X \cup \alpha$  with support= $X.count$  **if**  $C(\beta) = true$
- 5         build the  $\beta$  conditional database and the corresponding FP tree  $Tree_{\beta}$
- 6         **if**  $Tree_{\beta}$  is not empty **then** call FP-grow( $Tree_{\beta}, \beta, minsup, C$ )

# FP Grow Algorithm with Monotone Constraint

Run the FP Grow algorithm with the following database with minsup 2 and constraint  $\text{sum}(S) \geq 5$  where the item price coincides with the item id.

Tid	Items
1	1, 3, 4
2	2, 3, 5
3	1, 2, 3, 5
4	2, 5



Item	Conditional database
2	-
3	2:2
5	2:1, 2,3:2
1	3:1, 2,3,5:1

5-conditional database

Tid	Items
1	2
2	2, 3
3	2, 3

Now  $C(5) = \text{true}$  so we replace  $C$  with  $C_{\text{true}}$ .

# FP Grow Algorithm with Antimonotone Constraint

**Algorithm:** FP-tree( $D$ ,  $minsup$ ,  $C$ )

**Input:** A transactional database  $D$ ,  $minsup$ , and an antimonotone constraint  $C$ .

**Output:** The FP tree for  $D$ ,  $minsup$  and  $C$ .

- 1 Count support for each item in  $D$
- 2 Remove the infrequent items from the transactions in  $D$
- 3 **Remove the items that do not satisfy  $C$  from the transactions in  $D$**
- 4 Sort the items in each transaction in  $D$  in support descending order
- 5 Create a FP tree with a single node  $T$  with  $T.name = NULL$
- 6 for each transaction  $I \in D$  do
- 7     insert-tree( $I$ ,  $T$ )

**Algorithm:** FP-grow( $Tree$ ,  $\alpha$ ,  $minsup$ ,  $C$ )

**Input:** A FP tree  $Tree$ , an itemset  $\alpha$ ,  $minsup$ , and an antimonotone constraint  $C$ .

**Output:** All the itemsets in  $Tree$  that end with  $\alpha$ , have  $minsup$  and satisfy  $C$ .

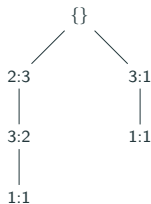
- 1 **let  $\delta$  denote all the items in  $Tree$**
- 2 **if  $C(\alpha \cup \delta) = true$  then**
- 3     **replace  $C$  with  $C_{true}$                     //  $C_{true}$  is a constraint that always returns true**
- 4 for each item  $X$  in  $Tree$  do
- 5     **if  $C(X \cup \alpha) = true$  then**
- 6         output the itemset  $\beta = X \cup \alpha$  with support= $X.count$
- 7         build the  $\beta$  conditional database and the corresponding FP tree  $Tree_\beta$
- 8         if  $Tree_\beta$  is not empty then call FP-grow( $Tree_\beta$ ,  $\beta$ ,  $minsup$ ,  $C$ )



# FP Grow Algorithm with Antimonotone Constraint

Run the FP Grow algorithm with the following database with minsup 2 and constraint  $\text{sum}(S) < 5$  where the item price coincides with the item id.

Tid	Items
1	1, 3, 4
2	2, 3, 5
3	1, 2, 3, 5
4	2, 5



Item	Conditional database
2	-
3	2:2
1	3:1, 2,3:1

3-conditional database

Tid	Items
1	2
2	2

Now  $C(3 \cup 2) = \text{false}$  so we stop.

## Convertible Monotone and Antimonotone Constraint

- We saw that  $\text{avg}(S) \leq v$  and  $\text{avg} \geq v$  was neither monotone nor antimonotone.
- A constraint  $C$  is convertible monotone when there exists an item order  $R$  such that for every itemsets  $A$  and  $B$  respecting  $R$  such that  $A$  is a suffix of  $B$ , if  $C(A) = \text{true}$  then  $C(B) = \text{true}$ .
  - $\text{avg}(S) \geq v$  with respect to decreasing price order.
- A constraint  $C$  is convertible antimonotone when there exists an item order  $R$  such that for every itemsets  $A$  and  $B$  respecting  $R$  such that  $B$  is a suffix for  $A$ , if  $C(A) = \text{true}$  then  $C(B) = \text{true}$ .
  - $\text{avg}(S) \geq v$  with respect to increasing price order.
- Alternatively,  $C$  is convertible antimonotone when there exists an item order  $R$  such that for every itemsets  $A$  and  $B$  respecting  $R$  such that  $B$  is a suffix for  $A$ , if  $C(B) = \text{false}$  then  $C(A) = \text{false}$ .
- A constraint that is both convertible monotone and antimonotone is called strongly convertible.

# Convertible Monotone and Antimonotone Constraints

Constraint	Convertible antimonotone	Convertible monotone	Strongly convertible
$\text{avg}(S) \leq, \geq v$	Yes	Yes	Yes
$\text{median}(S) \leq, \geq v$	Yes	Yes	Yes
$\text{sum}(S) \leq v^1, v \geq 0$	Yes	No	No
$\text{sum}(S) \leq v^1, v \leq 0$	No	Yes	No
$\text{sum}(S) \geq v^1, v \geq 0$	No	Yes	No
$\text{sum}(S) \geq v^1, v \leq 0$	Yes	No	No
$\vdots$	$\vdots$	$\vdots$	$\vdots$

---

<sup>1</sup>For the sum constraints the prices are of any value (negative or positive)

- Constraints can be added to the mining process to find itemsets that satisfy a certain constraint.
- Constraints can be **antimonotone**, **monotone**, **convertible** or **inconvertible**
- Depending on the type of constraint different modification to the mining algorithms are made.