Lecture 2
Week 2 (March 17) and Week 3 (March 24)

33459-01 Principles of Knowledge Discovery in Data

Association Rule Mining

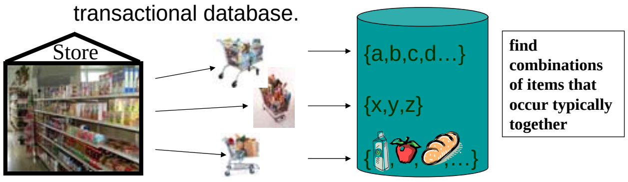
Lecture by: Dr. Osmar R. Zaïane

Course Content

- Introduction to Data Mining
- Association Analysis
- Sequential Pattern Analysis
- Classification and Prediction
- Contrast Sets
- Data Clustering
- Outlier Detection
- Web Mining

What Is Association Rule Mining?

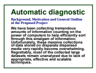
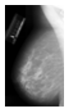
- **Association rule mining searches for relationships between items in a dataset**:
  – aims at discovering associations between items in a transactional database.



- Rule form: "**Body → Head [support, confidence]**"

buys(x, "bread") → buys(x, "milk") [0.6%, 65%]
major(x, "CS") ^ takes(x, "DB") → grade(x, "A") [1%, 75%]

Transactional Databases

	Transaction	Frequent itemset	Rule
	{bread, milk, beer,…}	(Bread, milk)	Bread → milk
	{term$_1$, term$_2$,…,term$_n$}	(term$_2$, term$_{25}$)	term2 → term25
	{f1, f2,…,Ca}	(f3, f5, fα)	f3^f5 → fα

Association Rule Mining

- **mining association rules** (Agrawal et. al SIGMOD93)
- **Fast algorithm** (Agrawal et. al VLDB94)
- **Partitioning** (Navathe et. al VLDB95)
- **Hash-based** (Park et. al SIGMOD95)
- **Multilevel A.R.** (Han et. al. VLDB95)
- **Generalized A.R.** (Srikant et. Al. VLDB95)
- **Quantitative A.R.** (Srikant et. al SIGMOD96)
- **Incremental mining** (Cheung et. al ICDE96)
- **Parallel mining** (Agrawal et. al TKDE96)
- **Distributed mining** (Cheung et. al PDIS96)
- **Meta-ruleguided mining** (Kamber et al. KDD97)
- **Direct Itemset Counting** (Brin et. al SIGMOD97)
- **N-dimensional A.R.** (Lu et. al DMKD'98)
- **Constraint A.R.** (Ng et. al SIGMOD'98)
- **A.R. with recurrent items** (Zaïane et. al ICDE'00)
- **FP without Candidate gen.** (Han et. al SIGMOD'00)
- **DualMiner** (Bucil, et. al KDD'02)
- **COFI algorithm** (El-Hajj, et. al Dawak'03)

And many many others:
Spatial AR; Sequence Associations;AR for multimedia; AR in time series;AR with progressive refinement; etc.

Lecture Outline

Part I: Concepts *(30 minutes)*
- **Basic concepts**
  - **Support and Confidence**
- **Naïve approach**

Part II: The Apriori Algorithm *(30 minutes)*
- **Principles**
- **Algorithm**
- **Running Example**

Part III: The FP-Growth Algorithm *(30 minutes)*
- **FP-tree structure**
- **Running Example**

Part IV: More Advanced Concepts *(30 minutes)*
- **Database layout and space search approach**
- **Other types of patterns and constraints**

Finding Rules in Transaction Data Set

- **6 transactions**
- **5 items: {Beer, Bread, Jelly, Milk, PeanutButter}**

Transactions	Items
T1	Bread, Jelly, PeanutButter
T2	Bread, PeanutButter
T3	Bread, Milk, PeanutButter
T4	Beer, Bread
T5	Beer, Milk
T6	Bread, Milk

- **Searching for rules of the form X→Y, where X and Y are sets of items**
  - **e.g. Bread → Jelly; Bread, Jelly → PeanutButter**
- **Design an efficient algorithm for mining association rules in large data sets**
- **Develop an effective approach for distinguishing interesting rules from irrelevant ones**

Basic Concepts

A transaction is a set of items: $T=\{i_a, i_b,\ldots i_t\}$

$T \subset I$, where I is the set of all possible items $\{i_1, i_2,\ldots i_d\}$

D, the task relevant data, is a set of transactions $D=\{T_1, T_2,\ldots T_n\}$.

An association rule is of the form:
P →Q, where $P \subset I$, $Q \subset I$, and $P \cap Q = \varnothing$

Basic Concepts (con't)

A set of items is referred to as itemset.

An itemset containing k items is called **k-itemset**.
{Jelly, Milk, Bread} is a 3-itemset example

An items set can also be seen as a conjunction of items (or a predicate)

P→Q holds in *D* with support ***s***
and
P→Q has a confidence ***c*** in the transaction set *D*.

Support(P→Q) = Probability(P∪Q)
Confidence(P→Q) = Probability(Q/P)

Support of an Itemset

- **Support** of $P = P_1 \wedge P_2 \wedge \ldots \wedge P_k$ in ***D*** σ(P/*D*) is the probability that P occurs in D: it is the percentage of transactions T in *D* satisfying P.
- **I.e. the *support of an item (or itemset) X* is the percentage of transactions in which that item (or items) occurs:** (number of T by cardinality of *D*).

$$\text{support}(X) = \frac{\#X}{n}$$

- **Support for all subsets of items**
  - **Note the exponential growth in the set of items**
  - **– 5 items: 31 sets**

Transactions	Items
T1	Bread, Jelly, PeanutButter
T2	Bread, PeanutButter
T3	Bread, Milk, PeanutButter
T4	Beer, Bread
T5	Beer, Milk
T6	Bread, Milk

Itemset	Support	Itemset	Support
Beer	33%	Beer, Bread, Milk	0%
Bread	66%	Beer, Bread, PeanutButter	0%
Jelly	16%	Beer, Jelly, Milk	0%
Milk	50%	Beer, Jelly, PeanutButter	0%
PeanutButter	50%	Beer, Milk, PeanutButter	0%
Beer, Bread	16%	Bread, Jelly, Milk	0%
Beer, Jelly	0%	Bread, Jelly, PeanutButter	16%
Beer, Milk	16%	Bread, Milk, PeanutButter	16%
Beer, PeanutButter	0%	Jelly, Milk, PeanutButter	0%
Bread, Jelly	16%	Beer, Bread, Jelly, Milk	0%
Bread, Milk	33%	Beer, Bread, Jelly, PeanutButter	0%
Bread, PeanutButter	50%	Beer, Bread, Milk, PeanutButter	0%
Jelly, Milk	0%	Beer, Jelly, Milk, PeanutButter	0%
Jelly, PeanutButter	16%	Bread, Jelly, Milk, PeanutButter	0%
Milk, PeanutButter	16%	Beer, Bread, Jelly, Milk, PeanutButter	0%
Beer, Bread, Jelly	0%		

Support and Confidence of an Association Rule

- **The *support of an association rule X→Y* is the percentage of transactions that contain X ∪Y**

$$\text{support}(X->Y) = \frac{\#(X \cup Y)}{n}$$

- **The *confidence of an association rule X→Y* is the ratio of the number of transactions that contain X ∪Y to the number of transactions that contain X**

$$\text{confidence}(X->Y) = \frac{\#(X \cup Y)}{\#X}$$

- **Confidence** of a rule P → Q in database D φ(P → Q/ *D*) is the ratio σ((P ∧ Q)/ *D*) by σ(P/ *D*)

$$\text{confidence}(X->Y) = \frac{\text{support}(X->Y)}{\text{support}(X)}$$

Support and Confidence – cont.

- **What is the support and confidence of the following rules?**
  - **Beer→Bread**
  - **{Bread, PeanutButter}→Jelly**

Transactions	Items
T1	Bread, Jelly, PeanutButter
T2	Bread, PeanutButter
T3	Bread, Milk, PeanutButter
T4	Beer, Bread
T5	Beer, Milk
T6	Bread, Milk

- **Support and confidence for some association rules**

Rule	Support	Confidence
Bread → PeanutButter	50%	75%
PeanutButter → Bread	50%	100%
Beer → Bread	16%	50%
PeanutButter → Jelly	16%	33%
Jelly → PeanutButter	16%	100%
Jelly → Milk	0%	0%
{Bread, PeanutButter} → Jelly	16%	33%

Why the difference?

- **Support measures how often the rule occurs in the database.**
- **Confidence measures the strength of the rule.**

Frequent Itemsets and Strong Rules

Support and Confidence are bound by Thresholds:

- *minimum support σ'*
- *minimum confidence φ'*

A **Frequent (or large) itemset** I in D is an itemset with a support larger than the minimum support;
A **strong rule** X→Y is a rule that is frequent (i.e. support higher than minimum support) and its confidence is higher than the minimum confidence threshold.

Association Rule Problem Definition

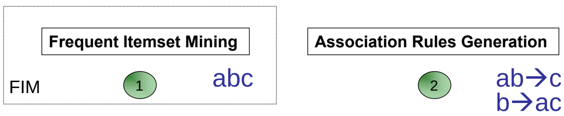
- Given $I=\{i_1, i_2, \ldots, i_m\}$, $D=\{t_1, t_2, \ldots, t_n\}$ and the support and confidence thresholds, the *association rule mining problem* is to identify **all strong** association rules X→Y.

Naïve Approach to Generate Association Rules

- **Enumerate all possible rules and select those of them that satisfy the minimum support and confidence thresholds**
- **Not practical for large databases**
  - **For a given dataset with *m* items, the total number of possible rules is $3^m-2^{m+1}+1$**
  - **For our example: $3^5-2^6+1= 180$**
  - **More than 80% of these rules are discarded if σ'=0.2 and φ'=0.5**
- **We need a strategy for rule generation - generate only the promising rules**

Better Approach

- Find the *frequent itemsets*: the sets of items that have minimum support
- Use the frequent itemsets to generate association rules. Keep only strong rules.



Bound by a **support** threshold   Bound by a **confidence** threshold

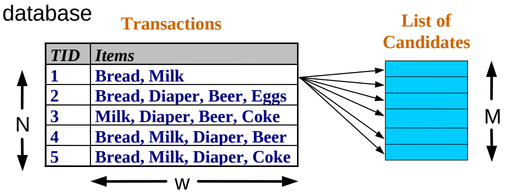
Generating Association Rules from Frequent Itemsets

- Only strong association rules are generated.
- Frequent itemsets satisfy minimum support threshold.
- Strong AR satisfy minimum confidence threshold.
- $\text{Confidence}(A \rightarrow B) = \text{Prob}(B/A) = \dfrac{\text{Support}(A \cup B)}{\text{Support}(A)}$

For each frequent itemset, **f**, generate all non-empty subsets of **f**.
For every non-empty subset **s** of **f** **do**
  output rule **s→(f-s)** if support(**f**)/support(**s**) ≥ min_confidence
end

Naïve Frequent Itemset Generation

- Brute-force approach (Basic approach):
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database



TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

  - Match each transaction against every candidate
  - Complexity ~ O(NMw) => **Expensive since $M = 2^d$** !!!

Lecture Outline

Part I: Concepts *(30 minutes)*
- **Basic concepts**
  - **Support and Confidence**
- **Naïve approach**

Part II: The Apriori Algorithm *(30 minutes)*
- **Principles**
- **Algorithm**
- **Running Example**

Part III: The FP-Growth Algorithm *(30 minutes)*
- **FP-tree structure**
- **Running Example**

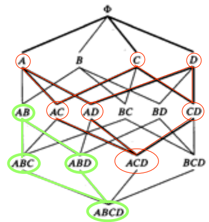
Part IV: More Advanced Concepts *(30 minutes)*
- **Database layout and space search approach**
- **Other types of patterns and constraints**

An Influential Mining Methodology — The *Apriori* Algorithm

- The *Apriori* method:
  - Proposed by Agrawal & Srikant 1994
  - A similar level-wise algorithm by Mannila et al. 1994
- Major idea (*Apriori Principle*):
  - A subset of a frequent itemset must be frequent
    - E.g., if **{beer, diaper, nuts}** is frequent, **{beer, diaper}** must be. Any itemset that is infrequent, its superset cannot be frequent!
  - **A powerful, scalable candidate set pruning technique:**
    - **It reduces candidate k-itemsets dramatically (for k > 2)**

Apriori Algorithm

- **Apriori principle:**
  - ***A subset of any frequent (large) itemset is also frequent***
  - **This also implies that *if an itemset is not frequent (small), a superset of it is also not frequent***
    - **If we know that an itemset is infrequent, we need not generate any subsets of it as they will be infrequent**



- **Lines represent "subset" relationship**
- **If ACD is frequent, than AC,AD,CD,A,C,D are also frequent, i.e. if an itemset is frequent than any set in a path above it is also frequent**
- **If AB is infrequent, than ABC, ABD, ABCD will also be infrequent, i.e. if an itemset is infrequent than any set in the path below is also infrequent**
- **If any of A, C, D, AC, AD, CD, is infrequent than ACD is infrequent (no need to check).**

Mining Association rules: the Key Steps

- Find the *frequent itemsets*: the sets of items that have minimum support
  - ◆A subset of a frequent itemset must also be a frequent itemset, i.e., if {*AB*} is a frequent itemset, both {*A*} and {*B*} should be frequent itemsets
  - ◆Iteratively find frequent itemsets with cardinality from 1 to *k* (*k*-itemsets*)*
- Use the frequent itemsets to generate strong association rules.

Apriori Algorithm – Idea

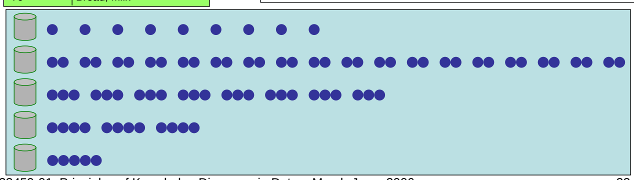
- **1. Generate candidate itemsets of a particular size**
- **2. Scan the database to see which of them are frequent**
  - **An itemset is frequent if all its subsets are frequent**
- **3. Use only these frequent itemsets to generate the set of candidates with size=size+1**

For our example if σ'=50%

Transactions	Items
T1	Bread, Jelly, PeanutButter
T2	Bread, PeanutButter
T3	Bread, Milk, PeanutButter
T4	Beer, Bread
T5	Beer, Milk
T6	Bread, Milk

itemset size

Pass	Candidates	Frequent itemsets
1	{Beer}, {Bread}, {Jelly}, {Milk}, {PeanutButter}	{Bread}(66%), {Milk}(50%) {PeanutButter}(50%)
2	{Bread, Milk}, {Bread, PeanutButter} {Milk, PeanutButter}	{Bread, PeanutButter}(50%)



The Apriori Algorithm

C_k: Candidate itemset of size k
L_k : frequent itemset of size k

L_1 = {frequent items};
for (k = 1; L_k !=∅; k++) **do begin**
  C_{k+1} = candidates generated from L_k;
  for each transaction *t* in database **do**
    increment the count of all candidates in C_{k+1} that are contained in *t*
  L_{k+1} = candidates in C_{k+1} with min_support
end
return $\cup_k L_k$;

The Apriori Algorithm -- Example

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D → C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

Support > 1 → L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Scan D → C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

L_2

itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

itemset
{2 3 5}

Scan D → L_3

itemset	sup
{2 3 5}	2

Note: {1,2,3}{1,2,5} and {1,3,5} not in C_3

Apriori-Gen Algorithm – Clothing Example

- **Given: 20 clothing transactions; *s*=20%, *c*=50%**
- **Generate association rules using the Apriori algorithm**

Transaction	Items	Transaction	Items
t_1	Blouse	t_{11}	TShirt
t_2	Shoes, Skirt, TShirt	t_{12}	Blouse, Jeans, Shoes, Skirt, TShirt
t_3	Jeans, TShirt	t_{13}	Jeans, Shoes, Shorts, TShirt
t_4	Jeans, Shoes, TShirt	t_{14}	Shoes, Skirt, TShirt
t_5	Jeans, Shorts	t_{15}	Jeans, TShirt
t_6	Shoes, TShirt	t_{16}	Skirt, TShirt
t_7	Jeans, Skirt	t_{17}	Blouse, Jeans, Skirt
t_8	Jeans, Shoes, Shorts, TShirt	t_{18}	Jeans, Shoes, Shorts, TShirt
t_9	Jeans	t_{19}	Jeans
t_{10}	Jeans, Shoes, TShirt	t_{20}	Jeans, Shoes, Shorts, TShirt

- **Scan1: Find all 1-itemsets. Identify the frequent ones.**

  Candidates: Blouse, Jeans, Shoes, Shorts, Skirt, Tshirt
  Support: 3/20 14/20 10/20 5/20 6/20 14/20
  Frequent (Large): Jeans, Shoes, Shorts, Skirt, Tshirt
  Join the frequent items – combine items with each other to generate candidate pairs

Clothing Example – cont.1

- **Scan2: 10 candidate 2-itemsets were generated. Find the frequent ones.**

  {Jeans, Shoes}:7/20 {Shoes, Short}:4/20 {Short, Skirt}: 0/20 {Skirt, TShirt}: 4/20
  {Jeans, Short} :5/20 {Shoes, Skirt}: 3/20 {Short, TShirt}: 4/20
  {Jeans, Skirt} :3/20 {Shoes, TShirt}: 10/20
  {Jeans, TShirt}:9/20 4/20 *7 frequent itemsets are found out of 10.*

Scan	Candidates	Large Itemsets
1	{Blouse}, {Jeans}, {Shoes}, {Shorts}, {Skirt}, {TShirt}	{Jeans}, {Shoes}, {Shorts} {Skirt}, {Tshirt}
2	{Jeans, Shoes}, {Jeans, Shorts}, {Jeans, Skirt}, {Jeans, TShirt}, {Shoes, Shorts}, {Shoes, Skirt}, {Shoes, TShirt}, {Shorts, Skirt}, {Shorts, TShirt}, {Skirt, TShirt}	{Jeans, Shoes}, {Jeans, Shorts}, {Jeans, TShirt}, {Shoes, Shorts}, {Shoes, TShirt}, {Shorts, TShirt}, {Skirt, TShirt}
3	{Jeans, Shoes, Shorts}, {Jeans, Shoes, TShirt}, {Jeans, Shorts, TShirt}, {Jeans, Skirt, TShirt}, {Shoes, Shorts, TShirt}, {Shoes, Skirt, TShirt}, {Shorts, Skirt, TShirt}	{Jeans, Shoes, Shorts}, {Jeans, Shoes, TShirt}, {Jeans, Shorts, TShirt}, {Shoes, Shorts, TShirt}
4	{Jeans, Shoes, Shorts, TShirt}	{Jeans, Shoes, Shorts, TShirt}
5	∅	∅

Everyone is combined with each other

2 sets are joined if they have 1 item in common (i,.e. 1 item different)

2 sets are joined if they have 2 item in common (i,.e. 1 item different)

Clothing Example – cont.2

- **The next step is to use the large itemsets and generate association rules**
- **c=50%**
- **The set of large itemsets is**

  L={{Jeans},{Shoes}, {Shorts}, {Skirt}, {TShirt}, {Jeans, Shoes}, {Jeans, Shorts}, {Jeans, TShirt}, {Shoes, Shorts}, {Shoes, TShirt}, {Shorts, TShirt}, {Skirt, TShirt}, {Jeans, Shoes, Shorts}, {Jeans, Shoes, TShirt}, {Jeans, Shorts, TShirt},{Shoes, Shorts, TShirt}, {Jeans, Shoes, Shorts,TShirt} }

- **We ignore the first 5 as they do not consists of 2 nonempty subsets of large itemsets. We test all the others, e.g.:**

$$confidence(Jeans->Shoes) = \frac{\text{support}(\{Jeans, Shoes\})}{\text{support}(\{Jeans\})} = \frac{7/20}{14/20} = 50\% \geq c$$

etc.

Lecture Outline

Part I: Concepts *(30 minutes)*
- **Basic concepts**
  - **Support and Confidence**
- **Naïve approach**

Part II: The Apriori Algorithm *(30 minutes)*
- **Principles**
- **Algorithm**
- **Running Example**

Part III: The FP-Growth Algorithm *(30 minutes)*
- **FP-tree structure**
- **Running Example**

Part IV: More Advanced Concepts *(30 minutes)*
- **Database layout and space search approach**
- **Other types of patterns and constraints**

Problems with Apriori

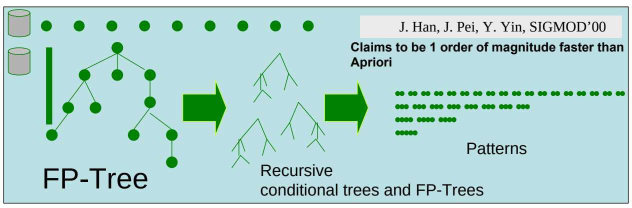
- Generation of candidate itemsets are expensive (Huge candidate sets)
  - 10^4 frequent 1-itemset will generate 10^7 candidate 2-itemsets
  - To discover a frequent pattern of size 100, e.g., $\{a_1, a_2, \ldots, a_{100}\}$, one needs to generate $2^{100} \approx 10^{30}$ candidates.
- High number of data scans

Frequent Pattern Growth

- First algorithm that allows frequent pattern mining without generating candidate sets
- Requires Frequent Pattern Tree

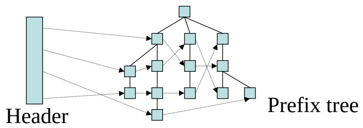
FP-Growth

- Grow long patterns from short ones using local frequent items
  - "abc" is a frequent pattern
  - Get all transactions having "abc": DB|abc
  - "d" is a local frequent item in DB|abc → abcd is a frequent pattern

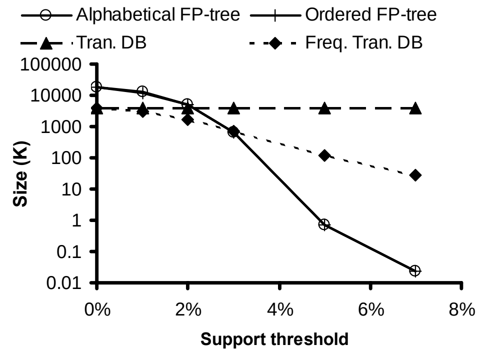


Frequent Pattern Tree

- Prefix tree.
- Each node contains the item name, frequency and pointer to another node of the same kind.
- Frequent item header that contains item names and pointer to the first node in FP tree.



Database Compression Using FP-tree (on T10I4D100k)



Frequent Pattern Tree

F, A, C, D, G, I, M, P
A, B, C, F, L, M, O
B, F, H, J, O
A, F, C, E, L, P, M, N
B, C, K, S, P
F, M, C, B, A

Required Support: **3**

F:5, C:5, A:4, B:4, M:4, P:3 ~~D:1 E:1 G:1 H:1 I:1 J:1 K:1 L:1 O:1~~

Frequent Pattern Tree

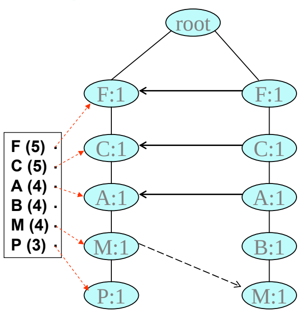
Original Transaction	Ordered frequent items
F, A, C, D, G, I, M, P	F, C, A, M, P
A, B, C, F, L, M, O	F, C, A, B, M
B, F, H, J, O	F, B
A, F, C, E, L, P, M, N	C, B, P
B, C, K, S, P	F, C, A, M, P
F, M, C, B, A	F, C, A, M
F, B, D	F, B

F:5, C:5, A:4, B:4, M:4, P:3

Required Support: **3**

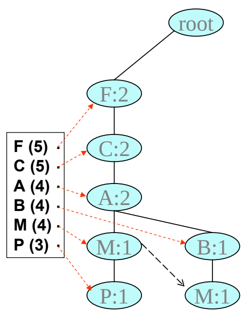
Frequent Pattern Tree

F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B



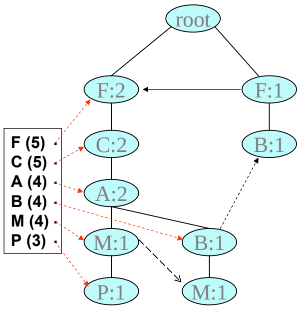
Frequent Pattern Tree

F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B



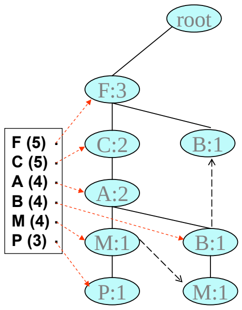
Frequent Pattern Tree

F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B



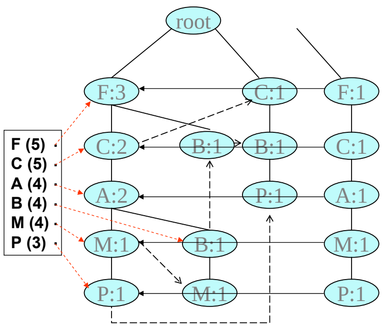
Frequent Pattern Tree

F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B



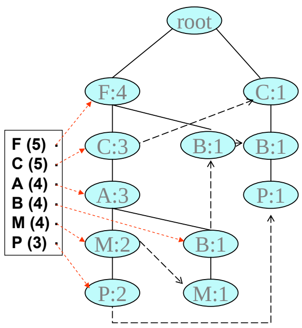
Frequent Pattern Tree

F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B



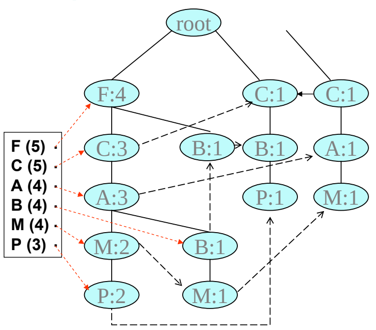
Frequent Pattern Tree

F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B



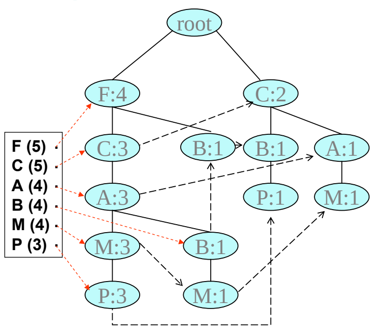
Frequent Pattern Tree

F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B



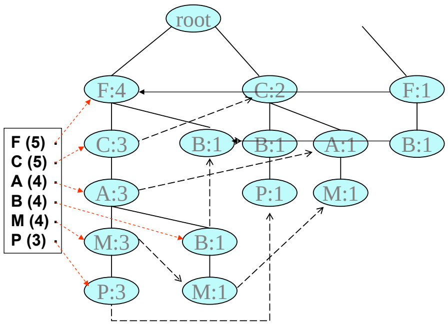
Frequent Pattern Tree

F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B



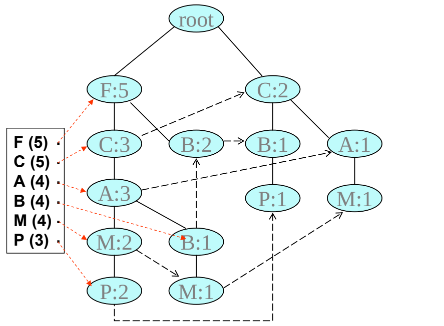
F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B

Frequent Pattern Tree



F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B

Frequent Pattern Tree



Mining Frequent Patterns with FP-Tree 3 Major Steps

Starting the processing from the end of list L:

Step 1:

Construct conditional pattern base for each item in the header table

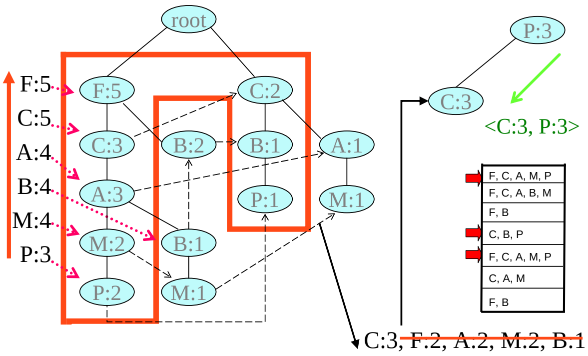
Step 2

Construct conditional FP-tree from each conditional pattern base

Step 3

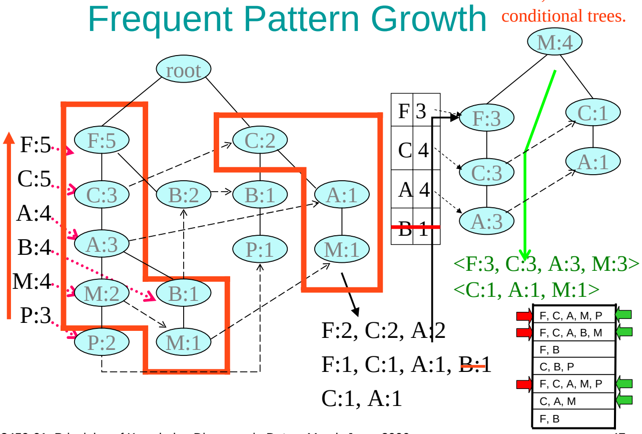
Recursively mine conditional FP-trees and grow frequent patterns obtained so far. If the conditional FP-tree contains a single path, simply enumerate all the patterns

Frequent Pattern Growth



Frequent Pattern Growth

Recursively build the A, C and F conditional trees.



Another Example: Construct FP-tree from a Transaction Database

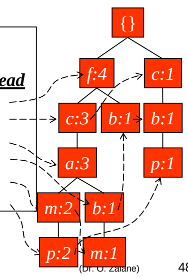
TID	***Items bought***	***(ordered) frequent items***
100	***{f, a, c, d, g, i, m, p}***	***{f, c, a, m, p}***
200	***{a, b, c, f, l, m, o}***	***{f, c, a, b, m}***
300	***{b, f, h, j, o, w}***	***{f, b}***
400	***{b, c, k, s, p}***	***{c, b, p}***
500	***{a, f, c, e, l, p, m, n}***	***{f, c, a, m, p}***

min_support = 3

1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Sort frequent items in frequency descending order, F-List
3. Scan DB again, construct FP-tree

Header Table

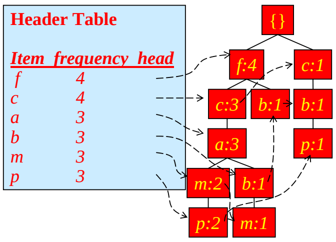
Item	*frequency*	*head*
f	*4*	
c	*4*	
a	*3*	
b	*3*	
m	*3*	
p	*3*	



F-list=f-c-a-b-m-p

Step 1: Construct Conditional Pattern Base

- Starting at the frequent-item header table in the FP-tree
- Traverse the FP-tree by following the link of each frequent item
- Accumulate all of transformed prefix paths of that item to form a conditional pattern base



Conditional pattern bases

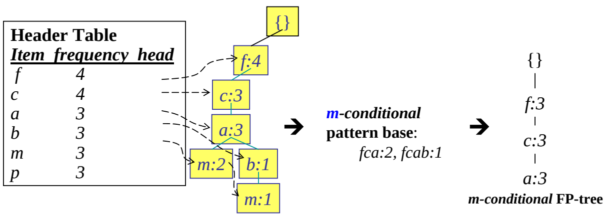
item	*cond. pattern base*
p	*fcam:2, cb:1*
m	*fca:2, fcab:1*
b	*fca:1, f:1, c:1*
a	*fc:3*
c	*f:3*

Properties of Step 1

- Node-link property
  - For any frequent item a_i, all the possible frequent patterns that contain a_i can be obtained by following a_i's node-links, starting from a_i's head in the FP-tree header.
- Prefix path property
  - To calculate the frequent patterns for a node a_i in a path P, only the prefix sub-path of a_i in P need to be accumulated, and its frequency count should carry the same count as node a_i.

Step 2: Construct Conditional FP-tree

- For each pattern base
  - Accumulate the count for each item in the base
  - Construct the conditional FP-tree for the frequent items of the pattern base

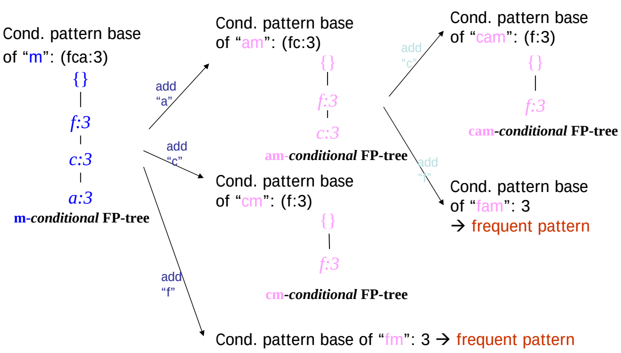


Conditional Pattern Bases and Conditional FP-Tree

Item	Conditional pattern base	Conditional FP-tree
p	{(fcam:2), (cb:1)}	{(c:3)}|p
m	{(fca:2), (fcab:1)}	{(f:3, c:3, a:3)}|m
b	{(fca:1), (f:1), (c:1)}	Empty
a	{(fc:3)}	{(f:3, c:3)}|a
c	{(f:3)}	{(f:3)}|c
f	Empty	Empty

order of L

Step 3: Recursively mine the conditional FP-tree



Principles of FP-Growth

- Pattern growth property
  - Let α be a frequent itemset in DB, B be α's conditional pattern base, and β be an itemset in B. Then $\alpha \cup \beta$ is a frequent itemset in DB iff β is frequent in B.
- Is "*fcabm* " a frequent pattern?
  - "fcab" is a branch of m's conditional pattern base
  - "b" is **NOT** frequent in transactions containing "fcab"
  - "bm" is **NOT** a frequent itemset.

Single FP-tree Path Generation

- Suppose an FP-tree T has a single path P. The complete set of frequent pattern of T can be generated by enumeration of all the combinations of the sub-paths of P

{}
|
f:3
|
c:3
|
a:3

m-conditional FP-tree

→

All frequent patterns concerning *m*

m,
fm, cm, am,
fcm, fam, cam,
fcam

Discussion (1/2)

- **Association rules are typically sought for very large databases → efficient algorithms are needed**
- **The Apriori algorithm makes 1 pass through the dataset for each different itemset size**
  - **The maximum number of database scans is k+1, where k is the cardinality of the largest large itemset (4 in the clothing ex.)**
  - **potentially large number of scans – weakness of Apriori**
- **Sometimes the database is too big to be kept in memory and must be kept on disk**
- **The amount of computation also depends on the min.support; the confidence has less impact as it does not affect the number of passes**
- **Variations**
  - **Using sampling of the database**
  - **Using partitioning of the database**
  - **Generation of incremental rules**

Discussion (2/2)

- Choice of minimum support threshold
  - lowering support threshold results in more frequent itemsets
  - this may increase number of candidates and max length of frequent itemsets
- Dimensionality (number of items) of the data set
  - more space is needed to store support count of each item
  - if number of frequent items also increases, both computation and I/O costs may also increase
- Size of database
  - since Apriori makes multiple passes, run time of algorithm may increase with number of transactions
- Average transaction width
  - transaction width increases with denser data sets
  - This may increase max length of frequent itemsets and traversals of hash tree (number of subsets in a transaction increases with its width)

Lecture Outline

Part I: Concepts *(30 minutes)*
- **Basic concepts**
  - **Support and Confidence**
- **Naïve approach**

Part II: The Apriori Algorithm *(30 minutes)*
- **Principles**
- **Algorithm**
- **Running Example**

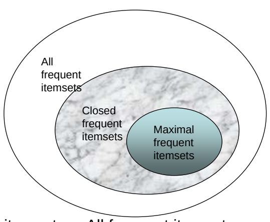
Part III: The FP-Growth Algorithm *(30 minutes)*
- **FP-tree structure**
- **Running Example**

Part IV: More Advanced Concepts *(30 minutes)*
- **Database layout and space search approach**
- **Other types of patterns and constraints**

Other Frequent Patterns

- Frequent pattern $\{a_1, \ldots, a_{100}\}$ → $\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100}-1 = 1.27*10^{30}$ frequent sub-patterns!

- Frequent Closed Patterns
- Frequent Maximal Patterns
- All Frequent Patterns



Maximal frequent itemsets ⊆ Closed frequent itemsets ⊆ All frequent itemset

Frequent Closed Patterns

- For frequent itemset X, if there exists no item y such that every transaction containing X also contains y, then X is a frequent closed pattern
- In other words, frequent itemset X is closed if there is no item y, not already in X, that always accompanies X in all transactions where X occurs.
- Concise representation of frequent patterns. Can generate all frequent patterns with their support from frequent closed ones.
- Reduce number of patterns and rules
- N. Pasquier et al. In ICDT'99

{abcd}
{abc}
{bd}

Transactions
Support = 2

a	2
b	3
c	2
d	2
ab	2
ac	2
bc	2
bd	2
abc	2

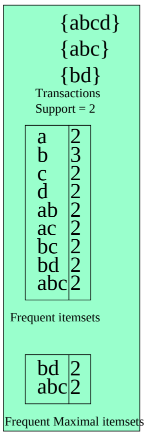
Frequent itemsets

b	3
bd	2
abc	2

Frequent Closed itemsets

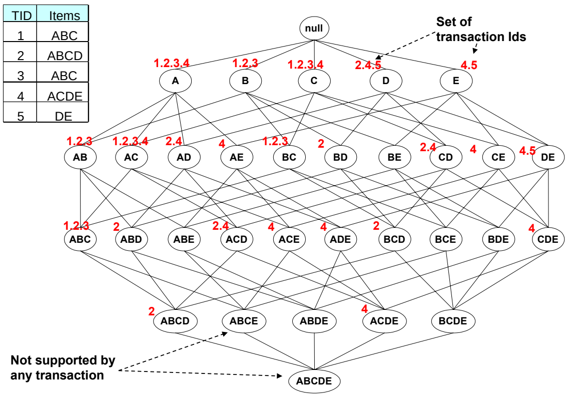
Frequent Maximal Patterns

- Frequent itemset X is maximal if there is no other frequent itemset Y that is superset of X.
- In other words, there is no other frequent pattern that would include a maximal pattern.
- More concise representation of frequent patterns but the information about supports is lost.
- Can generate all frequent patterns from frequent maximal ones but without their respective support.
- R. Bayardo. In SIGMOD'98



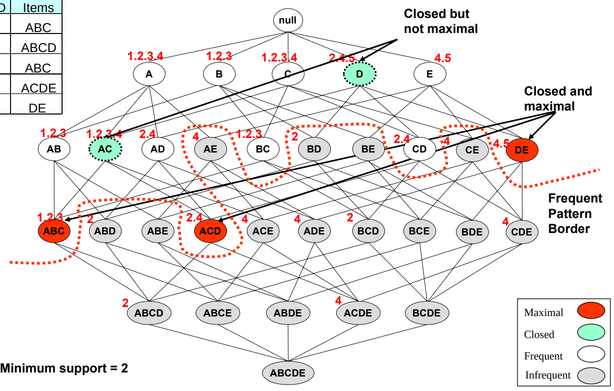
Maximal vs. Closed Itemsets

TID	Items
1	ABC
2	ABCD
3	ABC
4	ACDE
5	DE



Maximal vs. Closed Itemsets

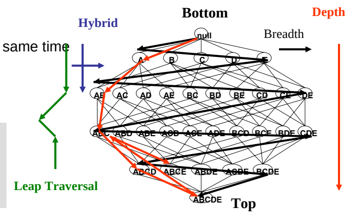
TID	Items
1	ABC
2	ABCD
3	ABC
4	ACDE
5	DE



Mining the Pattern Lattice

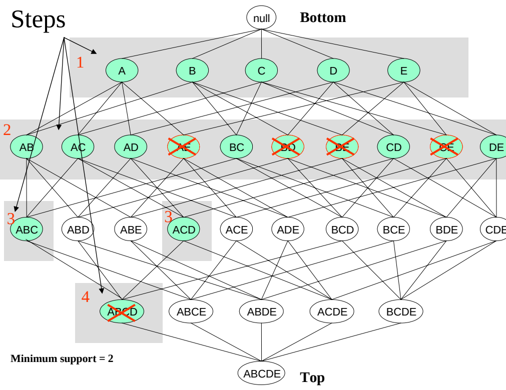
- Breadth-First
  - It uses current items at level k to generate items of level k+1 (many database scans)
- Depth-First
  - It uses a current item at level k to generate all its supersets (favored when mining long frequent patterns)
- Hybrid approach
  - It mines using both direction at the same time
- Leap traversal approach
  - Jumps to selected nodes

There is also the notion of :
Top-down (level k then level k+1)
Bottom-up (level k+1 then level k)



Breadth- First (Bottom-Up Example)

TID	Items
1	ABC
2	ABCD
3	ABC
4	ACDE
5	DE

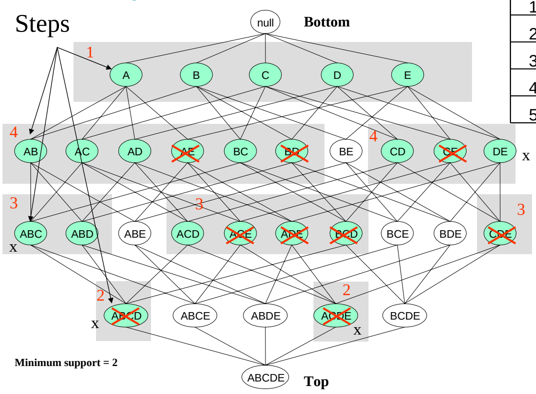


Superset is candidate if ALL its subsets are frequent

18 candidates to check

Depth First (Top-Down Example)

TID	Items
1	ABC
2	ABCD
3	ABC
4	ACDE
5	DE

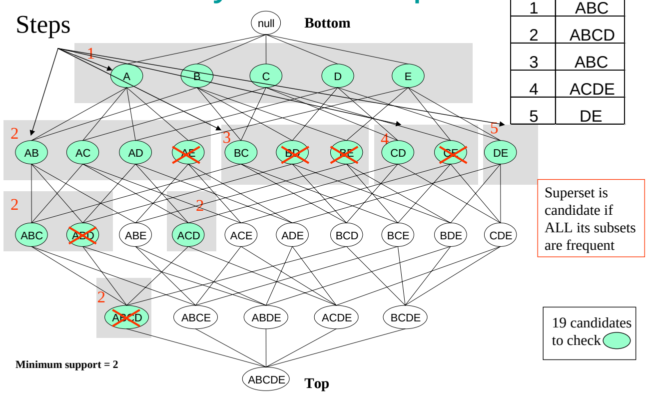


Subset is candidate if it is marked or if one of its supersets is candidate

23 candidates to check

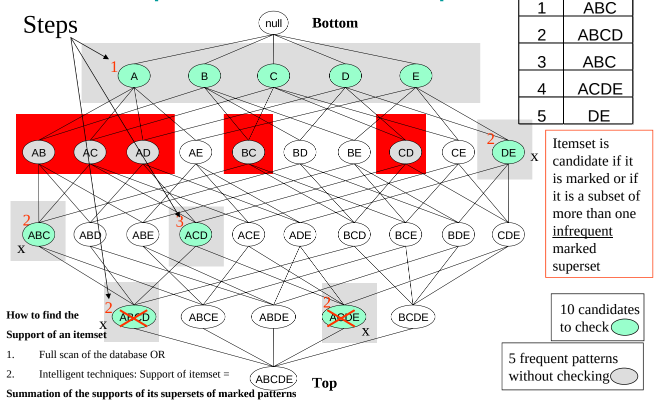
One Hybrid Example

TID	Items
1	ABC
2	ABCD
3	ABC
4	ACDE
5	DE



Leap Traversal Example

TID	Items
1	ABC
2	ABCD
3	ABC
4	ACDE
5	DE



Constraint-based Data Mining

- Finding all the patterns in a database autonomously? — unrealistic!
  - The patterns could be too many but not focused!
- Data mining should be an interactive process
  - User directs what to be mined using a data mining query language (or a graphical user interface)
- Constraint-based mining
  - User flexibility: provides constraints on what to be mined
  - System optimization: explores such constraints for efficient mining—**constraint-based mining**

Restricting Association Rules

- Useful for interactive and ad-hoc mining
- Reduces the set of association rules discovered and confines them to more relevant rules.
- **Before mining**
  - ✓ Knowledge type constraints: classification, etc.
  - ✓ Data constraints: SQL-like queries (DMQL)
  - ✓ Dimension/level constraints: relevance to some dimensions and some concept levels.
- **While mining**
  - ✓ Rule constraints: form, size, and content.
  - ✓ Interestingness constraints: support, confidence, correlation.
- **After mining**
  - ✓ Querying association rules

Constrained Frequent Pattern Mining: A Mining Query Optimization Problem

- Given a frequent pattern mining query with a set of constraints C, the algorithm should be
  - sound: it only finds frequent sets that satisfy the given constraints *C*
  - complete: all frequent sets satisfying the given constraints *C* are found
- A naïve solution
  - First find all frequent sets, and then test them for constraint satisfaction
- More efficient approaches:
  - Analyze the properties of constraints comprehensively
  - Push them as deeply as possible inside the frequent pattern computation.

Rule Constraints in Association Mining

- Two kind of rule constraints:
  - Rule form constraints: meta-rule guided mining.
    - P(x, y) ^ Q(x, w) –> takes(x, “database systems”).
  - Rule content constraint: constraint-based query optimization (where and having clauses) (Ng, et al., SIGMOD’98).
    - sum(LHS) < 100 ^ min(LHS) > 20 ^ count(LHS) > 3 ^ sum(RHS) > 1000
- **1-variable vs. 2-variable constraints** (Lakshmanan, et al. SIGMOD’99):
  - 1-var: A constraint confining only one side (L/R) of the rule, e.g., as shown above.
  - 2-var: A constraint confining both sides (L and R).
    - sum(LHS) < min(RHS) ^ max(RHS) < 5* sum(LHS)

Anti-Monotonicity in Constraint-Based Mining

TDB (min_sup=2)

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

- Anti-monotonicity
  - *When an intemset S* ***violates*** *the constraint, so does any of its supersets*
  - *sum(S.Price)* ≤ *v* is anti-monotone
  - *sum(S.Price)* ≥ *v* is not anti-monotone
- Example. C: range(S.profit) ≤ 15 is anti-monotone
  - Itemset *ab* violates C
  - So does every superset of *ab*

Monotonicity in Constraint-Based Mining

TDB (min_sup=2)

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

- Monotonicity
  - *When an intemset S* ***satisfies*** *the constraint, so does any of its supersets*
  - *sum(S.Price)* ≥ *v* is monotone
  - *min(S.Price)* ≤ *v* is monotone
- Example. C: range(S.profit) ≥ 15
  - Itemset *ab* satisfies C
  - So does every superset of *ab*

Which Constraints Are Monotone or Anti-Monotone?

SQL-based Constraints

Constraint	Monotone	Anti-Monotone
v ∈ S	yes	no
S ⊇ V	yes	no
S ⊆ V	no	yes
min(S) ≤ v	yes	no
min(S) ≥ v	no	yes
max(S) ≤ v	no	yes
max(S) ≥ v	yes	no
count(S) ≤ v	no	yes
count(S) ≥ v	yes	no
sum(S) ≤ v (a ∈ S, a ≤ 0)	no	yes
sum(S) ≥ v (a ∈ S, a ≤ 0)	yes	no
range(S) ≤ v	no	yes
range(S) ≥ v	yes	no
support(S) ≥ ξ	no	yes
support(S) ≤ ξ	yes	no

State Of The Art

- **Constraint pushing techniques have been proven to be effective in reducing the explored portion of the search space in constrained frequent pattern mining tasks.**
- **Anti-monotone constraints:**
  - **Easy to push …**
  - **Always profitable to do …**

FP-Growth with Constraints:
J. Pei, J. Han, L. Lakshmanan, ICDE'01

- **Monotone constraints:**
  - **Hard to push …**
  - **Should we push them, or not?**

- Dual Miner: C. Bucil, J. Gherke, D. Kiefer and W. White, SIGKDD'02
- FP-Bonsai: F. Bonchi anf B. Goethals, PAKDD'04
- COFI with constraints: M. El-Hajj and O. Zaiane, AI'05
- BifoldLeap: M. El-Hajj and O. Zaiane, ICDM'05