Ludwig-Maximilians-Universität München
Institut für Informatik
Lehr- und Forschungseinheit für Datenbanksysteme

Lecture notes

Knowledge Discovery in Databases

Summer Semester 2012

Lecture 3: Frequent Itemsets Mining & Association Rules Mining

Lecture: Dr. Eirini Ntoutsi
Exercises: Erich Schubert

http://www.dbs.ifi.lmu.de/cms/Knowledge_Discovery_in_Databases_I_(KDD_I)

Sources

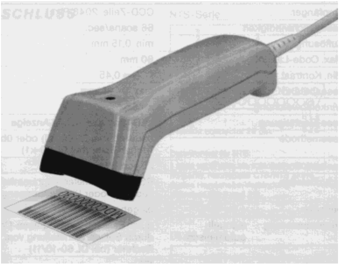
- Previous KDD I lectures on LMU (Johannes Aßfalg, Christian Böhm, Karsten Borgwardt, Martin Ester, Eshref Januzaj, Karin Kailing, Peer Kröger, Jörg Sander, Matthias Schubert, Arthur Zimek)
- Jiawei Han, Micheline Kamber and Jian Pei, *Data Mining: Concepts and Techniques, 3rd ed.,* Morgan Kaufmann, 2011.
- Margaret Dunham, Data Mining, *Introductory and Advanced Topics*, Prentice Hall, 2002.
- Wikipedia

Outline

- Introduction
- Basic concepts
- Frequent Itemsets Mining (FIM) – Apriori
- Association Rules Mining
- Apriori improvements
- Closed frequent itemsets (CFI) & Maximal frequent itemsets (MFI)
- Things you should know
- Homework/tutorial

Introduction

- Frequent patterns are patterns that appear frequently in a dataset.
  - Patterns: items, substructures, subsequences …
- Typical example: Market basket analysis



Customer transactions

Tid	Transaction items
1	Butter, Bread, Milk, Sugar
2	Butter, Flour, Milk, Sugar
3	Butter, Eggs, Milk, Salt
4	Eggs
5	Butter, Flour, Milk, Salt, Sugar

- We want to know: What products were often purchased together?
  - e.g.: beer and diapers?

The parable of the beer and diapers:
http://www.theregister.co.uk/2006/08/15/beer_diapers/

- Applications:
  - Improving store layout
  - Sales campaigns
  - Cross-marketing
  - Advertising

… its not only about market basket data

- Market basket analysis
  - Items are the products
  - Transactions are the products bought by a customer during a supermarket visit
  - Example: Buy(X, “Diapers”) $\rightarrow$ Buy(X, “Beer”) [0.5%, 60%]
- Similarly in an online shop, e.g. Amazon
  - Example: Buy(X, “Computer”) $\rightarrow$ Buy(X, “MS office”) [50%, 80%]
- University library
  - Items are the books
  - Transactions are the books borrowed by a student during the semester
- University
  - Items are the courses
  - Transactions are the courses that are chosen by a student
  - Example: Major (X, “CS”) $\wedge$ Course(X, “DB”) $\rightarrow$ grade(X, “A”) [1%, 75%]
- … and many other applications.
- Also, frequent patter mining is fundamental in other DM tasks.

Outline

- Introduction
- Basic concepts
- Frequent Itemsets Mining (FIM) – Apriori
- Association Rules Mining
- Apriori improvements
- Closed frequent itemsets (CFI) & Maximal frequent itemsets (MFI)
- Things you should know
- Homework/tutorial

Basic concepts: Items, itemsets and transactions

Tid	Transaction items
1	Butter, Bread, Milk, Sugar
2	Butter, Flour, Milk, Sugar
3	Butter, Eggs, Milk, Salt
4	Eggs
5	Butter, Flour, Milk, Salt, Sugar

- **Items I:** the set of items $I = \{i_1, \ldots, i_m\}$
  e.g. products in a supermarket, books in a bookstore
- **Itemset X**: A set of items $X \subseteq I$
- **Itemset size:** the number of items in the itemset
- ***k*-Itemset:** an itemset of size k
  e.g. {Butter, Brot, Milch, Zucker} is an 4-Itemset
  e.g. {Mehl, Wurst} is a 2-Itemset
- **Transaction T:** $T = (tid, X_T)$
  e.g. products bought during a customer visit to the supermarket
- ***Database DB***: A set of transactions T
  e.g. customers purchases in a supermarket during the last week
- Items in transactions or itemsets are lexicographically ordered

  Itemset $X = (x_1, x_2, \ldots, x_k)$, such as $x_1 \leq x_2 \leq \ldots \leq x_k$

Basic concepts: frequent itemsets, support

Tid	Transaction items
1	Butter, Bread, Milk, Sugar
2	Butter, Flour, Milk, Sugar
3	Butter, Eggs, Milk, Salt
4	Eggs
5	Butter, Flour, Milk, Salt, Sugar

Let X be an itemset.

- **Itemset cover:** the set of transactions containing *X*:

$$cover(X) = \{tid \mid (tid, X_T) \in DB, X \subseteq X_T\}$$

- **(absolute) support/ support count** of X**:** # transactions containing X

$$supportCount(X) = |cover(X)|$$

- **(relative) support** of X**:** the fraction of transactions that contain X (or the probability that a transaction contains X)

$$support(X) = \mathrm{P}(X) = supportCount(X) \,/\, |DB|$$

- **Frequent itemset**: An itemset X is frequent in DB if its support is no less than a minSupport threshold s:

$$support(X) \geq s$$

- **L_k: the set of frequent k-itemsets**
  - L comes from "Large" ("large itemsets"), another term for "frequent itemsets"

Example: itemsets

Tid	Transaction items
1	Butter, Bread, Milk, Sugar
2	Butter, Flour, Milk, Sugar
3	Butter, Eggs, Milk, Salt
4	Eggs
5	Butter, Flour, Milk, Salt, Sugar

I = {Butter, Bread, Eggs, Flour, Milk, Salt, Sugar}

- support(Butter) = 4/5=80%
  - cover(Butter) = {1,2,3,4}
- support(Butter, Bread) = 1/5=20%
  - cover(Butter, Bread) = ….
- support(Butter, Flour) = 2/5=40%
  - cover(Butter, Flour) = ….
- support(Butter, Milk, Sugar) = 3/5=60%
  - Cover(Butter, Milk, Sugar)= ….

Frequent Itemsets Mining (FIM) problem

Problem 1: Frequent Itemsets Mining (FIM)

Given:

- A set of items I
- A transactions database DB over I
- A *minSupport* threshold s

Goal: Find all frequent itemsets in *DB*, i.e.:

$$\{X \subseteq I \mid support(X) \geq s\}$$

TransaktionsID	Items
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Support of 1-Itemsets:

(A): 75%, (B), (C): 50%, (D), (E), (F): 25%,

Support of 2-Itemsets:

(A, C): 50%,

(A, B), (A, D), (B, C), (B, E), (B, F), (E, F): 25%

Basic concepts: association rules, support, confidence

Let X, Y be two itemsets: $X,Y \subseteq I$ *and* $X \cap Y = \emptyset$.

- **Association rules:** rules of the form

$$\boxed{X} \rightarrow \boxed{Y}$$

head or LHS (left-hand side) or antecedent of the rule *body or RHS (right-hand side) or consequent of the rule*

- **Support s** of a rule**:** the percentage of transactions containing $X \cup Y$ in the DB or the probability $P(X \cup Y)$

$$support(X \rightarrow Y)=P(X \cup Y) = support(X \cup Y)$$

- **Confidence c** of a rule**:** the percentage of transactions containing $X \cup Y$ in the set of transactions containing X or the conditional probability that a transaction containing X also contains Y

$$confidence(X \rightarrow Y)= P(Y|X)= P(X \cup Y)/P(X)=support(X \cup Y) / support (X)$$

- Support and confidence are measures of rules interestingness.
- Rules are usually written as follows:

$$X \rightarrow Y \text{ (support, confidence)}$$

Example: association rules

Tid	Transaction items
1	Butter, Bread, Milk, Sugar
2	Butter, Flour, Milk, Sugar
3	Butter, Eggs, Milk, Salt
4	Eggs
5	Butter, Flour, Milk, Salt, Sugar

I = {Butter, Bread, Eggs, Flour, Milk, Salt, Sugar}

Sample rules:

- Butter→Bread (20%, 25%)
  - support(Butter ∪Bread)=1/5=20%
  - support(Butter)=4/5=80%
  - Confidence = 20%/80%=1/4=25%
- {Butter, Milk} → Sugar (60%, 75%)
  - support(Butter, Milk ∪ Sugar) = 3/5=60%
  - Support(Butter,Milk) = 4/5=80%
  - Confidence = 60%/80%=3/4=75%

Problem 2: Association Rules Mining

Given:

- A set of items *I*
- A transactions database DB over *I*
- A *minSupport* threshold *s* and a *minConfidence* threshold *c*

Goal: Find all association rules $X \rightarrow Y$ in *DB* w.r.t. minimum support *s* and minimum confidence *c*, i.e.:

$$\{X \rightarrow Y \mid support(X \cup Y) \geq s, confidence(X \rightarrow Y) \geq c\}$$

These rules are called strong.

TransaktionsID	Items
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Association rules:

$A \Rightarrow C$ (Support = 50%, Confidence= 66.6%)

$C \Rightarrow A$ (Support = 50%, Confidence= 100%)

Problems solving

Problem 1 (FIM): Find all frequent itemsets in *DB*, i.e.: $\{X \subseteq I \mid support(X) \geq s\}$

Problem 2 (ARM): Find all association rules $X \rightarrow Y$ in *DB,* w.r.t. min support *s* and min confidence *c*: $\{X \rightarrow Y \mid support(X \cup Y) \geq s, confidence(X \rightarrow Y) \geq c, X,Y \subseteq I \text{ and } X \cap Y = \emptyset\}$

- Problem 1 is part of Problem 2:
  - Once we have support($X \cup Y$) and support(X), we can check if $X \rightarrow Y$ is strong.
- 2-step method to extract the association rules:
  1. Determine the frequent itemsets w.r.t. min support s: ⟵ FIM problem

     "Naïve" algorithm: count the frequencies for all *k*-itemsets

     Inefficient!!! There are $\binom{|I|}{k}$ such subsets

     Total cost: $O(2^{|I|})$

     => Apriori-algorithm and variants
  2. Generate the association rules w.r.t. min confidence *c*:

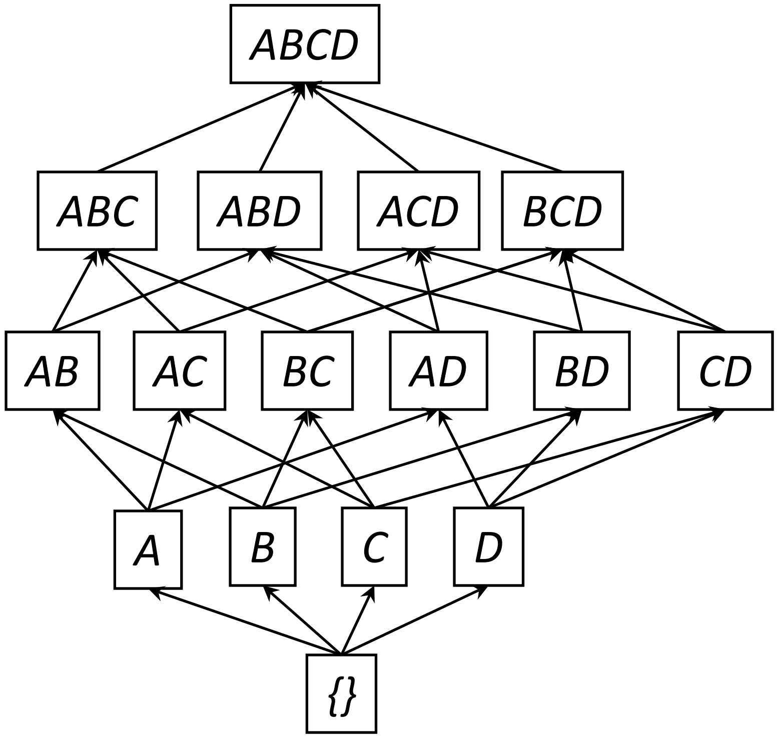
     from frequent itemset X, generate $Y \rightarrow (X - Y)$, $Y \subset X$, $Y \neq \emptyset$, $Y \neq X$

Step 1(FIM) is the most costly, so the overall performance of an association rules mining algorithm is determined by this step.

Itemsets lattice

- The number of itemsets can be really huge.

Let us consider a small set of items: I = {A,B,C,D}



- # 1-itemsets: $\binom{4}{1} = \frac{4!}{(4-1)!*1!} = \frac{4!}{3!} = 4$
- # 2-itemsets: $\binom{4}{2} = \frac{4!}{(4-2)!*2!} = \frac{4!}{2!*2!} = 6$
- # 3-itemsets: $\binom{4}{3} = \frac{4!}{(4-3)!*3!} = \frac{4!}{3!} = 4$
- # 4-itemsets: $\binom{4}{4} = \frac{4!}{(4-4)!*4!} = 1$

- In the general case, for |I| items, there exist:

$$\binom{|I|}{1} + \binom{|I|}{2} + \ldots + \binom{|I|}{k} = 2^{|I|} - 1$$

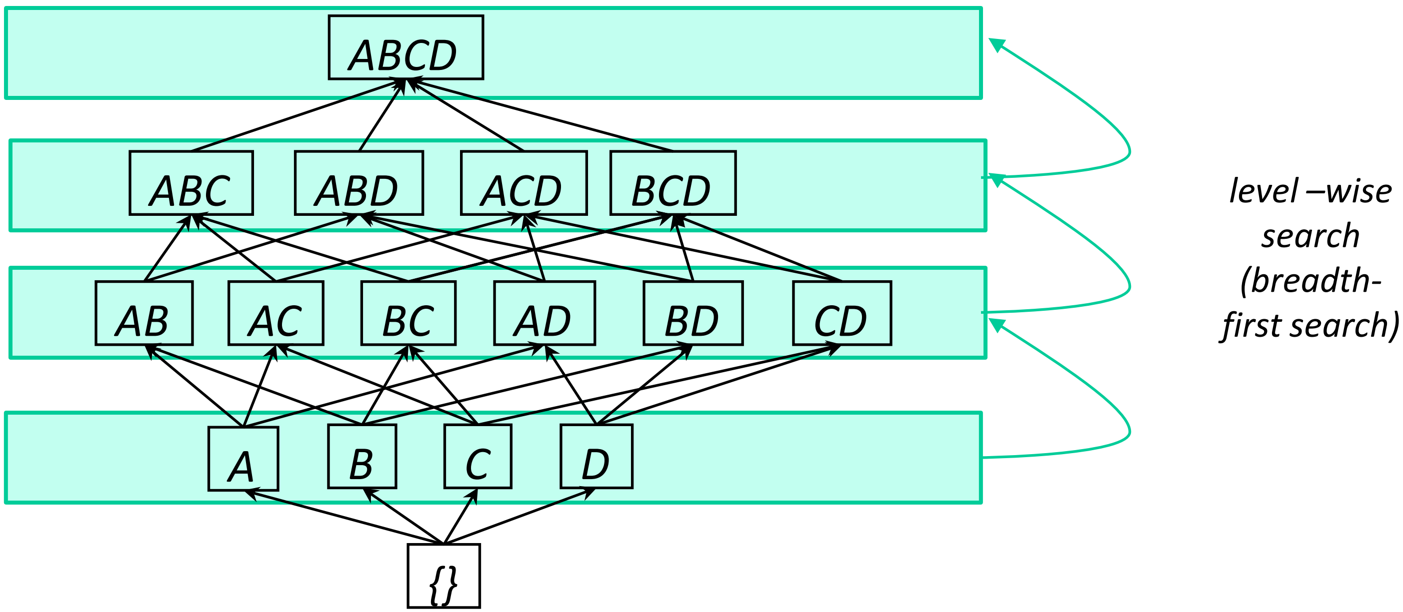
- So, generating all possible combinations and computing their support is inefficient!

Outline

- Introduction
- Basic concepts
- Frequent Itemsets Mining (FIM) – Apriori
- Association Rules Mining
- Apriori improvements
- Closed frequent itemsets (CFI) & Maximal frequent itemsets (MFI)
- Things you should know
- Homework/tutorial

Apriori algorithm [Agrawal & Srikant @VLDB'94]

- First, frequent 1-itemsets are determined, then frequent 2-itemsets and so on



- Method:
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Test the candidates against DB (one scan)
  - Terminate when no frequent or candidate set can be generated

Apriori property

- **Naïve approach:** Count the frequency of all k-itemsets from I

  test $\sum_{k=1}^{|I|} \binom{|I|}{k} = 2^{|I|} - 1$ itemsets, i.e., $O(2^{|I|})$.

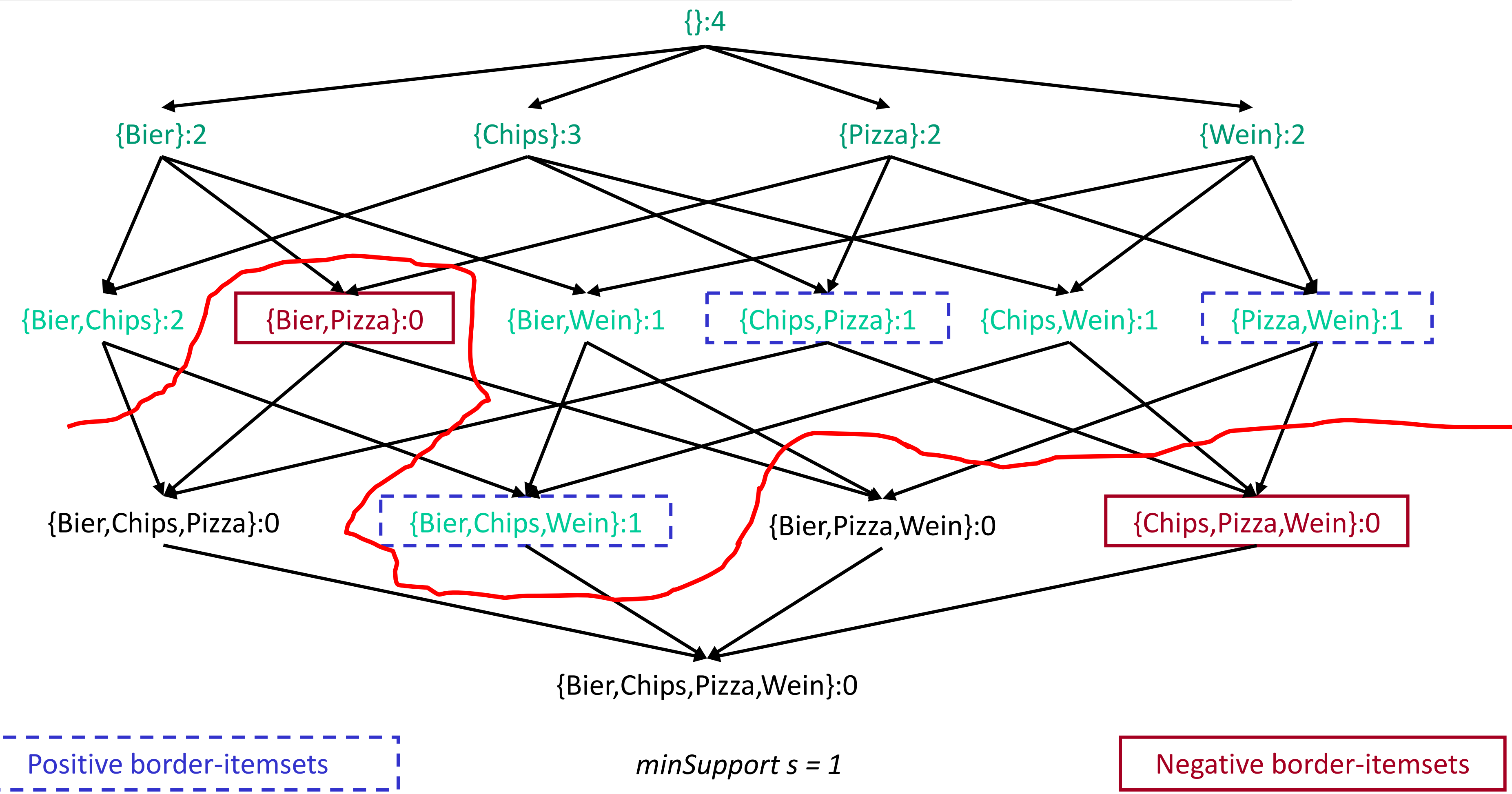
- Candidate itemset X:
  - the algorithm evaluates whether *X* is frequent
  - the set of candidates should be as small as possible!!!

- Downward closure property / Monotonic property of frequent itemsets:
  - If X is frequent, all its subsets $Y \subseteq X$ are also frequent.
    - If {beer, diaper, nuts} is frequent, so is {beer, diaper}
    - i.e., every transaction having {beer, diaper, nuts} also contains {beer, diaper}
    - similarly for {diaper, nuts}, {beer, nuts}
  - **Contrary:** When *X* is not frequent, all its supersets are not frequent and thus they should not be generated/ tested!!! → reduce the candidate itemsets set
    - If {beer, diaper} is not frequent, {beer, diaper, nuts} would not be frequent also

Search space

Border Itemset X: all subsets Y ⊂ X are frequent, all supersets Z ⊃ X are not frequent

- *Positive border: X is also frequent*
- *Negative border: X is not frequent*



From L_{k-1} to C_k to L_k

L_k: frequent itemsets of size k; C_k: candidate itemsets of size k

A 2-step process:

- **Join step:** generate candidates C_k
  - L_k is generated by self-joining L_{k-1}: $L_{k-1} * L_{k-1}$
  - Two (k-1)-itemsets p, q are joined, if they agree in the first (k-2) items
- **Prune step:** prune C_k and return L_k
  - C_k is superset of L_k
  - Naïve idea: count the support for all candidate itemsets in C_k ….$|C_k|$ might be large!
  - Use Apriori property: a candidate k-itemset that has some non-frequent (k-1)-itemset cannot be frequent
    - Prune all those k-itemsets, that have some (k-1)-subset that is not frequent (i.e. does not belong to L_{k-1})
    - Due to the level-wise approach of Apriori, we only need to check (k-1)-subsets
  - For the remaining itemsets in C_k, prune by support count (DB)

Example:
Let L_3=*{abc, abd, acd, ace, bcd}*

- Join step: $C_4=L_3*L_3$
C_4={abc*abd=abcd; acd*ace=acde}

- Prune step:
acde is pruned since cde is not frequent

Apriori algorithm (pseudo-code)

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

L_1 = {frequent items};
for (k = 1; L_k !=$\emptyset$; 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 $\cup_k L_k$;

Candidate generation (self-join, apriori property)

DB scan

subset function

Prune by support count

Subset function:
- The subset function must for every transaction T in DB check all candidates in the candidate set C_k whether they are part of the transaction T
- Organize candidates C_k in a hash tree

Example

$Sup_{min} = 2$

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

1st scan →

C_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

→ L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

← *2nd scan*

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

← L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

Itemset
{B, C, E}

3rd scan →

L_3

Itemset	sup
{B, C, E}	2

Apriori overview

- Advantages:
  - Apriori property
  - Easy implementation (in parallel also)
- Disadvantages:
  - It requires up to |I| database scans
    - |I| is the maximum transaction length
  - It assumes that the itemsets are in memory

Outline

- Introduction
- Basic concepts
- Frequent Itemsets Mining (FIM) – Apriori
- Association Rules Mining
- Apriori improvements
- Closed frequent itemsets (CFI) & Maximal frequent itemsets (MFI)
- Things you should know
- Homework/tutorial

Association Rules Mining

- (Recall the) 2-step method to extract the association rules:
  1. Determine the frequent itemsets w.r.t. min support s $\longleftarrow$ FIM problem (Apriori)
  2. Generate the association rules w.r.t. min confidence *c.*

- Regarding step 2, the following method is followed:
  - For every frequent itemset X
  - for every subset *Y* of *X: Y≠Ø, Y ≠X,* the rule *Y* → (*X* - *Y*) is formed
  - Remove rules that violate min confidence c

$$confidence(Y \rightarrow (X - Y)) = \frac{support\_count(X)}{support\_count(Y)}$$

  - Store the frequent itemsets and their supports in a hash table
    - no database scan!

Pseudocode

Input:

D //Database of transactions
I //Items
L //Large itemsets
s //Support
α //Confidence

Output:

R //Association Rules satisfying s *and* α

ARGen Algorithm:

```
R = ∅;
for each l ∈ L do
    for each x ⊂ l such that x ≠ ∅ and x ≠ l do
        if support(l)/support(x) ≥ α then
            R = R ∪ {x ⇒ (l − x)};
```

$$R = \emptyset;$$
for each $l \in L$ do
for each $x \subset l$ such that $x \neq \emptyset$ and $x \neq l$ do
if $\frac{support(l)}{support(x)} \geq \alpha$ then
$R = R \cup \{x \Rightarrow (l - x)\};$

Example

tid	X_T
1	{Bier, Chips, Wein}
2	{Bier, Chips}
3	{Pizza, Wein}
4	{Chips, Pizza}

Transaction database

I = {Bier, Chips, Pizza, Wein}

Itemset	Cover	Sup.	Freq.
{}	{1,2,3,4}	4	100 %
{Bier}	{1,2}	2	50 %
{Chips}	{1,2,4}	3	75 %
{Pizza}	{3,4}	2	50 %
{Wein}	{1,3}	2	50 %
{Bier, Chips}	{1,2}	2	50 %
{Bier, Wein}	{1}	1	25 %
{Chips, Pizza}	{4}	1	25 %
{Chips, Wein}	{1}	1	25 %
{Pizza, Wein}	{3}	1	25 %
{Bier, Chips, Wein}	{1}	1	25 %

Rule	Sup.	Freq.	Conf.
{Bier} ⇒ {Chips}	2	50 %	100 %
{Bier} ⇒ {Wein}	1	25 %	50 %
{Chips} ⇒ {Bier}	2	50 %	66 %
{Pizza} ⇒ {Chips}	1	25 %	50 %
{Pizza} ⇒ {Wein}	1	25 %	50 %
{Wein} ⇒ {Bier}	1	25 %	50 %
{Wein} ⇒ {Chips}	1	25 %	50 %
{Wein} ⇒ {Pizza}	1	25 %	50 %
{Bier, Chips} ⇒ {Wein}	1	25 %	50 %
{Bier, Wein} ⇒ {Chips}	1	25 %	100 %
{Chips, Wein} ⇒ {Bier}	1	25 %	100 %
{Bier} ⇒ {Chips, Wein}	1	25 %	50 %
{Wein} ⇒ {Bier, Chips}	1	25 %	50 %

Interesting and misleading association rules

Example:

- Database on the behavior of students in a school with 5000 students
- Itemsets:
  - 60% of the students play Soccer,
  - 75% of the students eat chocolate bars
  - 40% of the students play Soccer and eat chocolate bars
- Association rules:

"Play Soccer" → "Eat chocolate bars", confidence = 40%/60%= 67%

$\emptyset$ → "Eat chocolate bars", confidence= 75%

Playing Soccer and eating chocolate bars are negatively correlated

Task: Filter out misleading rules

- Condition for a rule $A \rightarrow B$

$$\frac{P(A \cup B)}{P(A)} > P(B) - d$$

  - for a suitable constant $d > 0$

- Measure of “interestingness“ of a rule: interest

$$\frac{P(A \cup B)}{P(A)} - P(B)$$

  - the higher the value the more interesting the rule is

- Measure of dependent/correlated events: lift

$$lift = \frac{P(A \cup B)}{P(A)P(B)}$$

  - the ratio of the observed support to that expected if X and Y were independent.

Measuring Quality of Association Rules

For a rule A→ B

- Support $P(A \cup B)$

  e.g. support(milk, bread, butter)=20%, i.e. 20% of the transactions contain these

- Confidence $\dfrac{P(A \cup B)}{P(A)}$

  e.g. confidence(milk, bread→ butter)=50%, i.e. 50% of the times a customer buys milk and bread, butter is bought as well.

- Lift $\dfrac{P(A \cup B)}{P(A)P(B)}$

  e.g. lift(milk, bread→ butter)=20%/(40%*40%)=1.25. the observed support is 20%, the expected (if they were independent) is 16%.

Outline

- Introduction
- Basic concepts
- Frequent Itemsets Mining (FIM) – Apriori
- Association Rules Mining
- Apriori improvements
- Closed frequent itemsets (CFI) & Maximal frequent itemsets (MFI)
- Things you should know
- Homework/tutorial

Apriori improvements

- Major computational challenges in Apriori:
  - Multiple scans of the DB: For each step (k-itemsets), a database scan is required
  - Huge number of candidates
  - Tedious workload of support counting for candidates
    - Too many candidates; One transaction may contain many candidates.
- Improving Apriori: general ideas
  - Reduce passes of transaction database scans
  - Shrink number of candidates
  - Facilitate support counting of candidates

Partition

Partition (A. Savasere, E. Omiecinski and S. Navathe, *VLDB'95)*

- Partition the DB into n non-overlapping partitions: DB_1, DB_2, …, DB_n
- Apply Apriori in each partition DB_i → extract *local frequent itemsets*
  - local minSupport threshold in DB_i: minSupport * $|DB_i|$

Pass 1

- Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB!
- The set of local frequent itemsets forms the *global candidate itemsets*
- Find the actual support of each candidate → *global frequent itemsets*

Pass 2

Partition cont'

Pseudocode

```
Divide D into partitions D¹,D²,…,Dᵖ;
For i = 1 to p do
    Lⁱ = Apriori(Dⁱ);                  // 1st pass
C = L¹ ∪ … ∪ Lᵖ;
Count C on D to generate L;            // 2nd pass
```

Advantages:

- adapted to fit in main memory size
- parallel execution
- 2 scans in DB

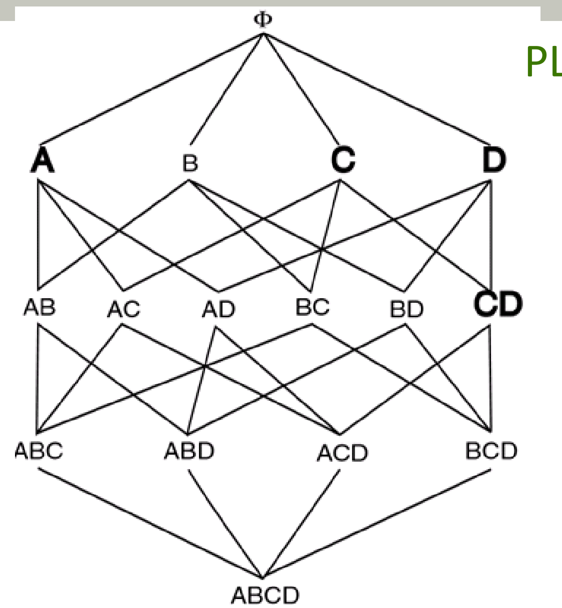
Dissadvantages:

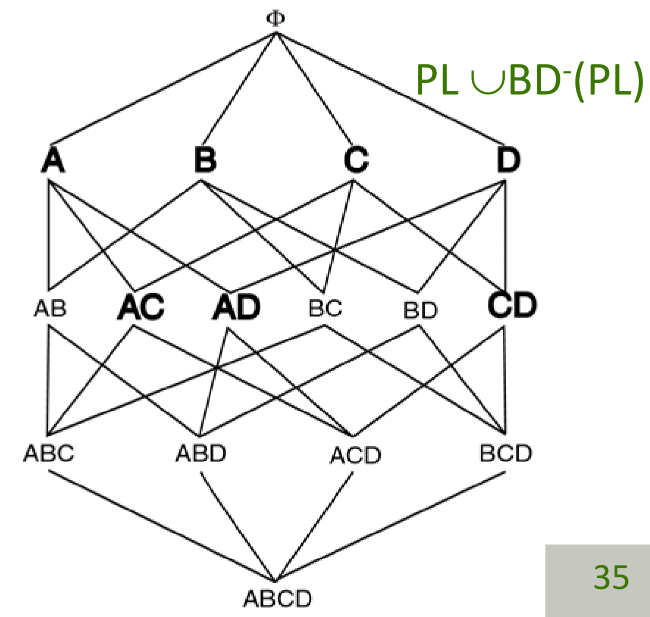
- # candidates in 2nd scan might be large

Sampling

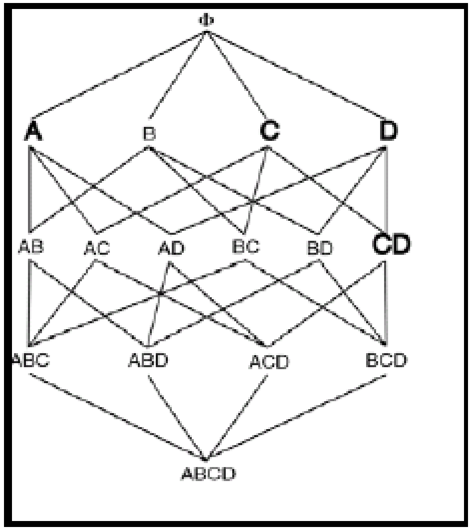
Sampling (H. Toinoven, *VLDB'96)*

- Select a sample S of DB (that fits in memory)
- Apply Apriori to the sample → PL *(potential large itemsets from sample)*
  - minSupport might be relaxed in the sample
- Since we search only in S, we might miss some global frequent itemsets
- Candidate set C = PL $\cup$BD$^-$(PL):
  - BD$^-$: negative border (minimal set of itemsets which are not in PL, but whose subsets are all in PL.)
- Count support of C in DB using minSupport
- If there are frequent itemsets in BD$^-$, expand C by repeatedly applying BD$^-$
- Finally, count C in DB

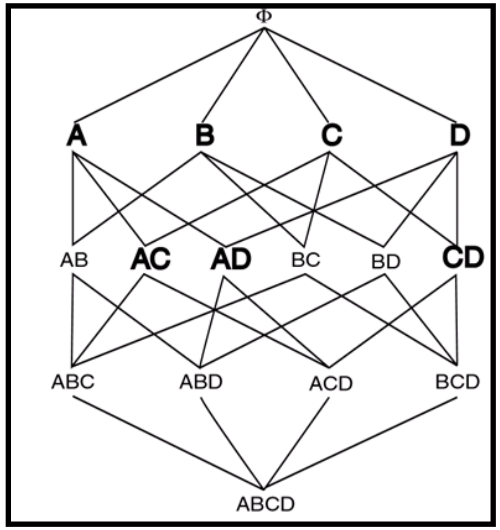




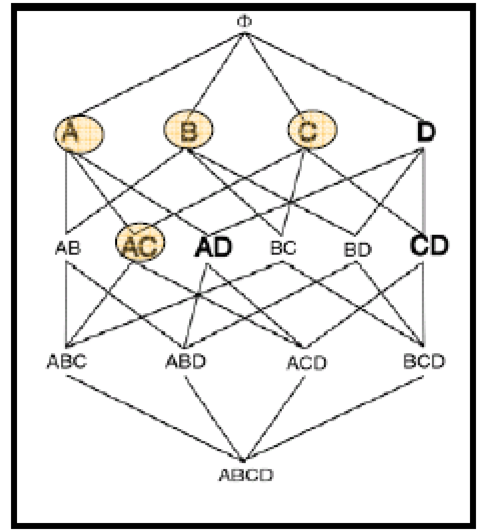
Sampling cont’



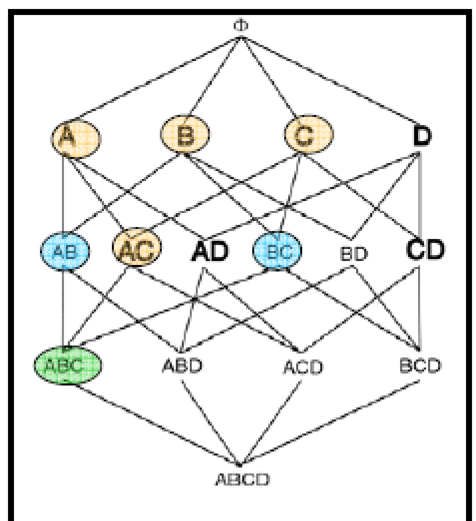
PL



$PL \cup BD^{-}(PL)$



frequent itemsets



Repeated application of BD^{-}

Sampling cont’

Pseudocode

```
D_s = sample of Database D;
PL = Large itemsets in D_s using smalls;
C = PL ∪ BD^-(PL);
Count C in Database using s;
ML = large itemsets in BD^-(PL);
If ML = ∅ then done
  else C = repeated application of BD^-;
          Count C in Database;
```

Advantages:

– Reduces number of database scans to 1 in the best case and 2 in worst.

– Scales better.

Disadvantages:

- The candidate set might be large

FPGrowth (Han, Pei & Yin, SIGMOD'00)

- Bottlenecks of the Apriori approach
  - Breadth-first (i.e., level-wise) search
  - Candidate generation and test
    - Often generates a huge number of candidates
- The FPGrowth (frequent pattern growth) approach
  - Depth-first search (DFS)
  - Avoid explicit candidate generation
  - Use an extension of prefix tree to "compact" the database

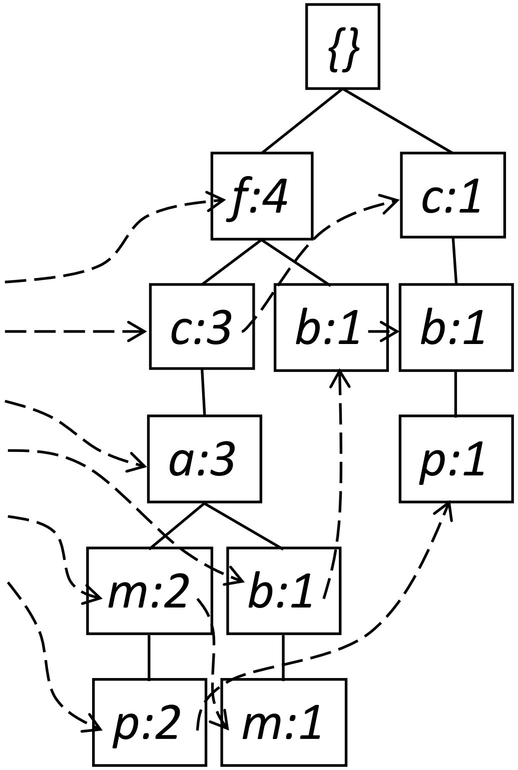
Construct FP-tree from transaction database

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

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

- To facilitate tree traversal, each item in the header table points to its occurrences in the tree via a chain of node-links
- most common items appear close to the root

FP-tree construction

- Frequent Pattern (FP) tree compresses the database, retaining the transactions information
- Method:
  1. Scan DB once, find frequent 1-itemset. Scan 1
  2. Sort frequent items in frequency descending order → f-list
  3. Scan DB again, construct FP-tree Scan 2
     - Create the root of the tree, labeled with “null”
     - Insert first transaction t1 in the tree e.g. t1=(A, B,C): create a new branch in the tree
     - Insert the next transaction t2 in the tree
       – If they are identical (i.e., t2=(A,B,C)), just update the the nodes along the path
       – If they share a common prefix (e.g., if t2=(A,B,D), update nodes in the shared part (A,B), create a new branch for the rest of the transaction (D)
       – If nothing in common, start a new branch from the root
     - Repeat for all transactions

Transaction items are accessed in f-list order!!!

Benefits of FP-tree structure

- Completeness
  - Preserve complete information for frequent pattern mining
  - Never break a long pattern of any transaction
- Compactness
  - Reduce irrelevant info—infrequent items are gone
  - Items in frequency descending order: the more frequently occurring, the more likely to be shared
  - Never be larger than the original database (not count node-links and the *count* field)
  - Achieves high compression ratio

Partition Patterns and Databases

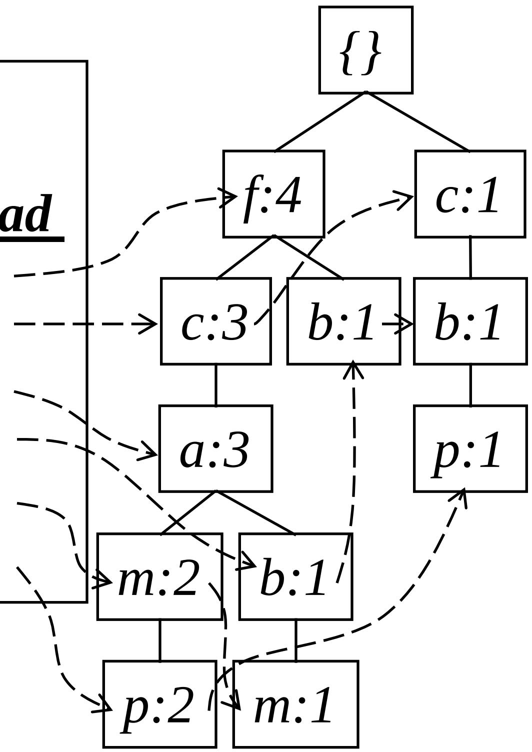
- Frequent patterns can be partitioned into subsets according to f-list
  - F-list=f-c-a-b-m-p
  - Patterns containing p
  - Patterns having m but no p
  - …
  - Patterns having c but no a nor b, m, p
  - Pattern f
- Completeness and non-redundency

Find Patterns Having P From P-conditional Database

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

Header Table

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



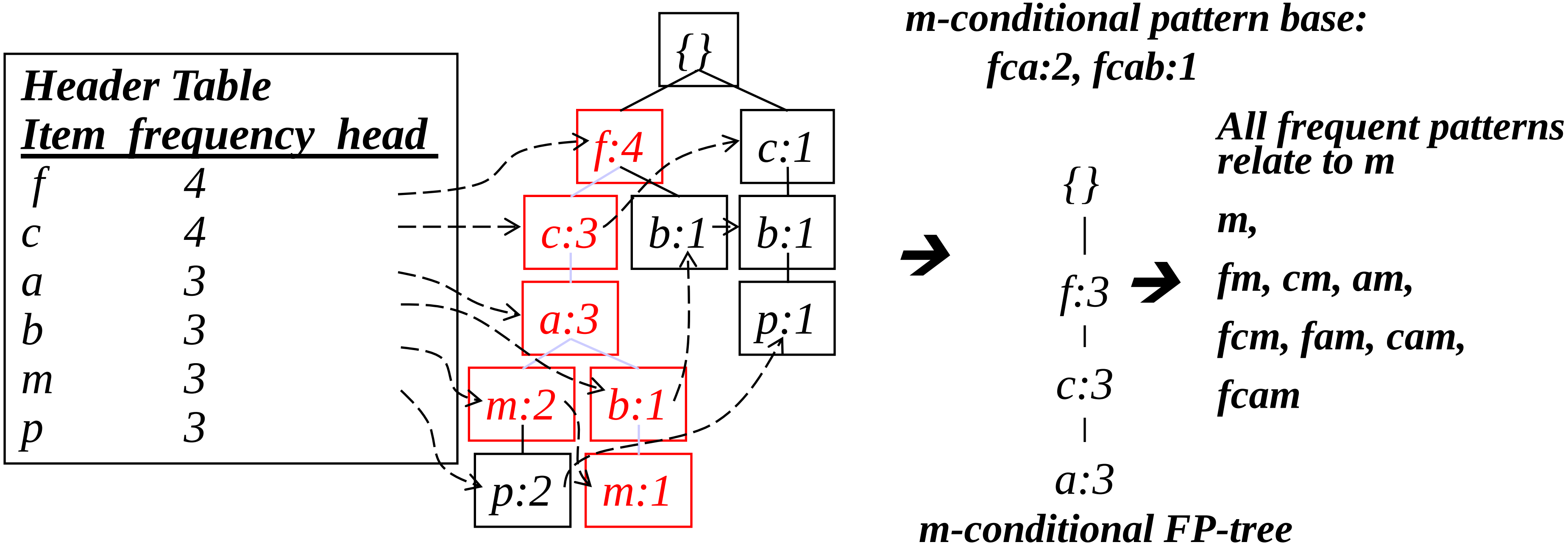
Conditional pattern bases

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

From Conditional Pattern-bases to Conditional FP-trees

For each pattern-base

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



Vertical vs horizontal representation

Transactions	A	B	C	D	E	F
100	1	1	0	0	0	1
200	1	0	1	1	0	0
300	0	1	0	1	1	1
400	0	0	0	0	1	1

Horizontal representation: {TID, itemset}

Items	100	200	300	400
A	1	1	0	0
B	1	0	1	0
C	0	1	0	0
D	0	1	1	0
E	0	0	1	1
F	1	0	1	1

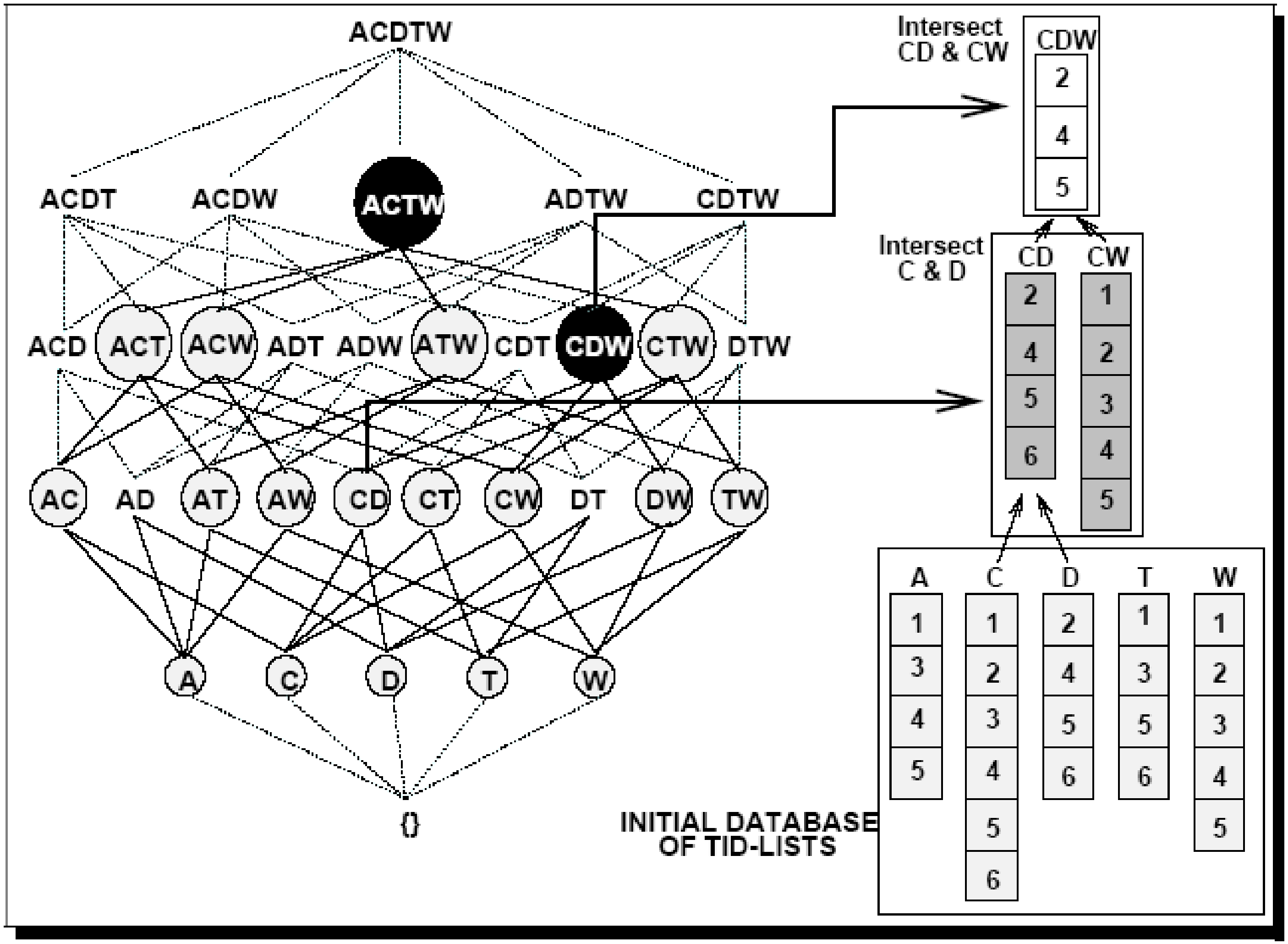
Vertical representation: {item, TID_set}

Mining with vertical representation

Eclat (Zaki, TKDE'00)

- Vertical data format
- For each itemset, a list of transaction ids containing the itemset is maintained
  - *X*.tidlist ={t1, t2, t3, t5}; Y.tidilist={t1, t2, t5, t8,t10}
- To find the support of X $\cup$Y, we use their lists intersection
  - *X*.tidlist $\cup$Y.tidilist={t1, t2, t5}
  - Support(X $\cup$Y)=| *X*.tidlist $\cup$Y.tidilist |=3
- No need to access the DB (use instead lists intersection)
- As we proceed, the size of the lists decrease, so intersection computation is faster

Example


ACDTW
ACDT
ACDW
ACTW
ADTW
CDTW
ACD
ACT
ACW
ADT
ADW
ATW
CDT
CDW
CTW
DTW
AC
AD
AT
AW
CD
CT
CW
DT
DW
TW
A
C
D
T
W
{}
Intersect CD & CW
CDW
2
4
5
Intersect C & D
CD
2
4
5
6
CW
1
2
3
4
5
INITIAL DATABASE OF TID-LISTS
A
1
3
4
5
C
1
2
3
4
5
6
D
2
4
5
6
T
1
3
5
6
W
1
2
3
4
5

Outline

- Introduction
- Basic concepts
- Frequent Itemsets Mining (FIM) – Apriori
- Association Rules Mining
- Apriori improvements
- Closed frequent itemsets (CFI) & Maximal frequent itemsets (MFI)
- Things you should know
- Homework/tutorial

To many frequent itemsets

- The number of frequent itemsets (FI) is too large
  - depends on the minSupport threshold of course, see example below

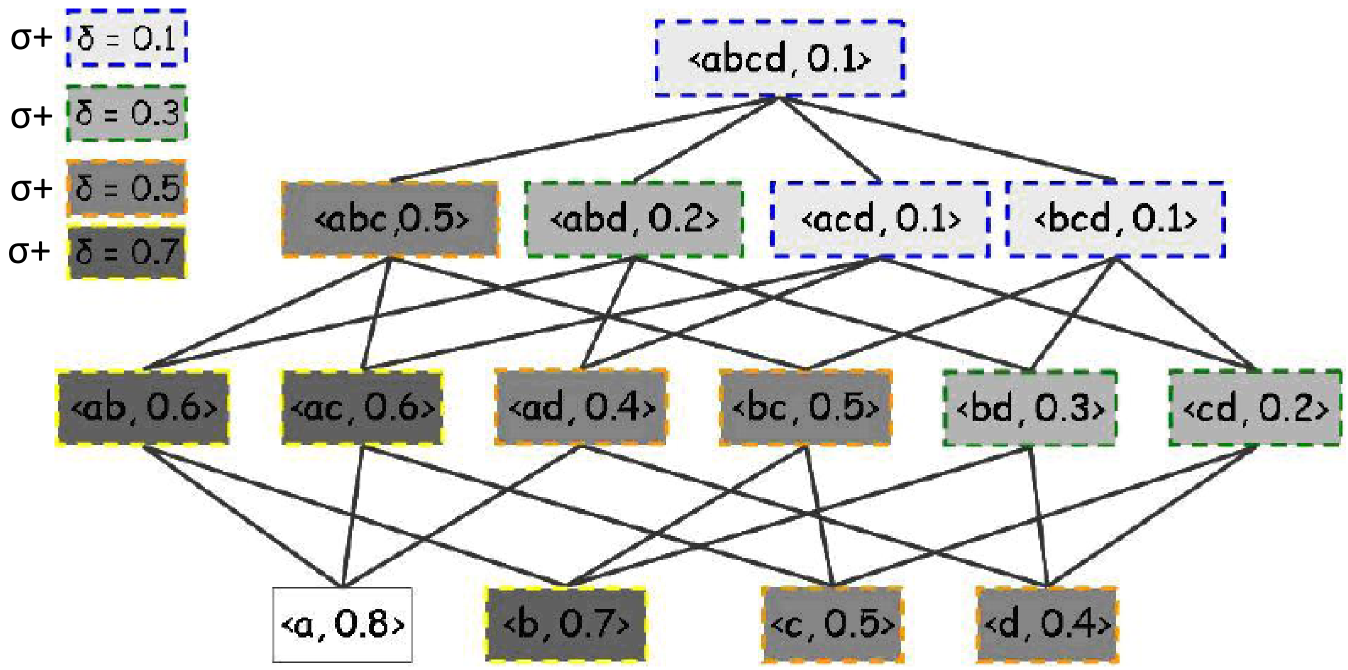
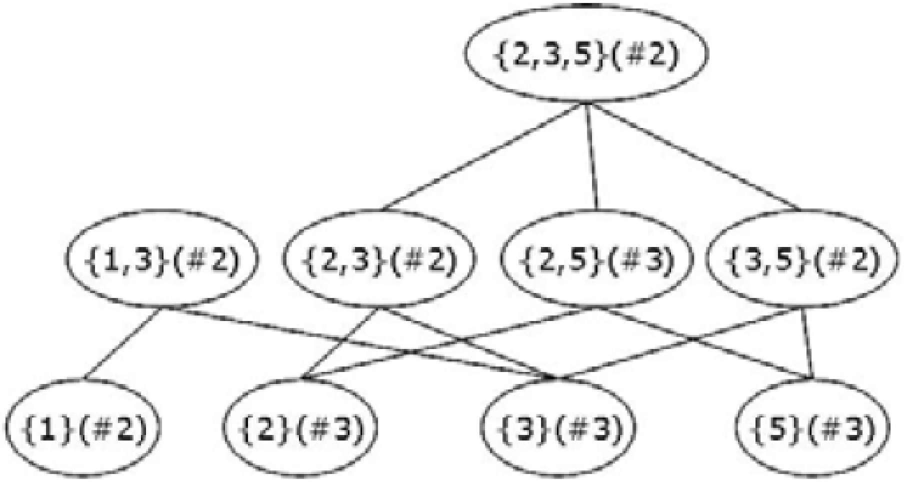


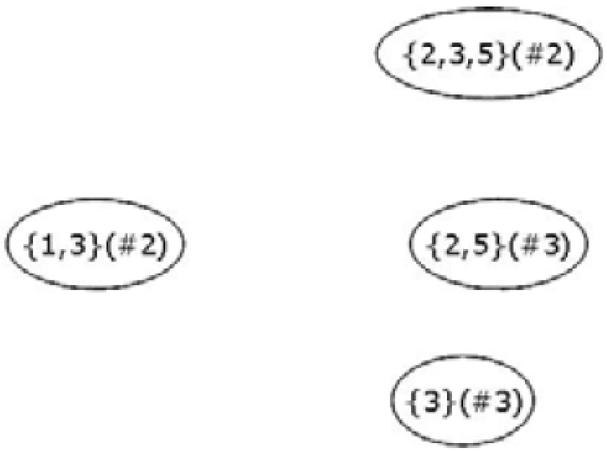
Figure 4.2: Effect of δ increase on the lattice structure ($\sigma = 0.1$)

Closed Frequent Itemsets (CFI)

- A frequent itemset X is called closed if there exists no frequent superset Y ⊇ X with suppD(X) = suppD(Y).
- The set of closed frequent itemsets is denoted by CFI
- CFIs comprise a lossless representation of the FIs since no information is lost, neither in structure (itemsets), nor in measure (support).



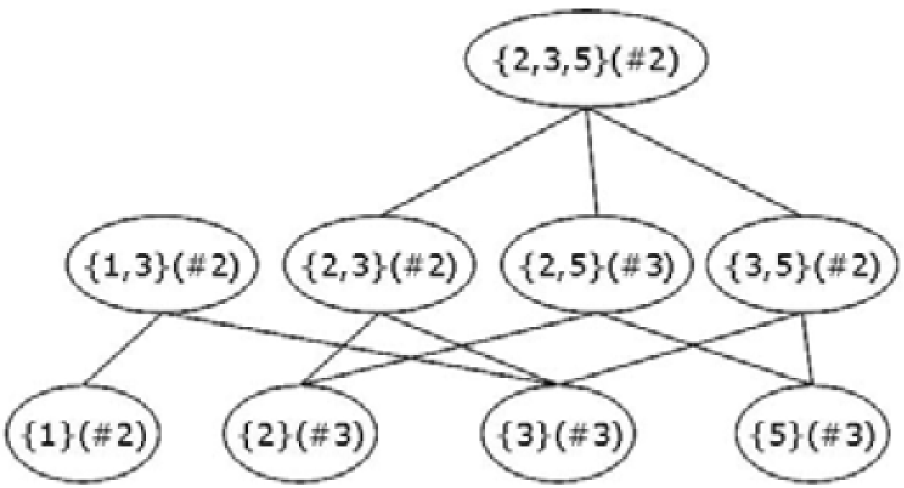
FI



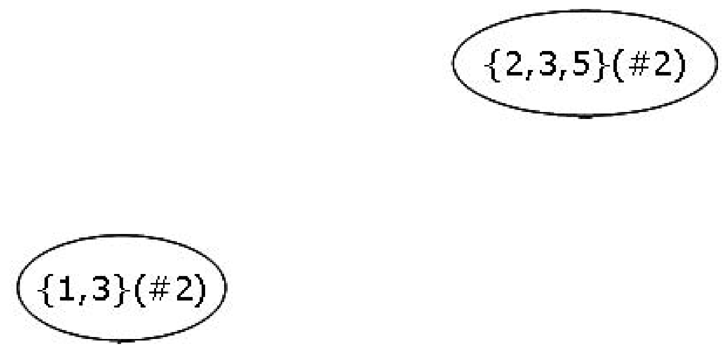
CFI

Maximal Frequent Itemsets (MFI)

- A frequent itemset is called maximal if it is not a subset of any other frequent itemset.
- The set of maximal frequent itemsets is denoted by MFI
- MFIs comprise a lossy representation of the FIs since it is only the lattice structure (i.e. frequent itemsets) that can be determined from MFIs whereas frequent itemsets supports are lost.

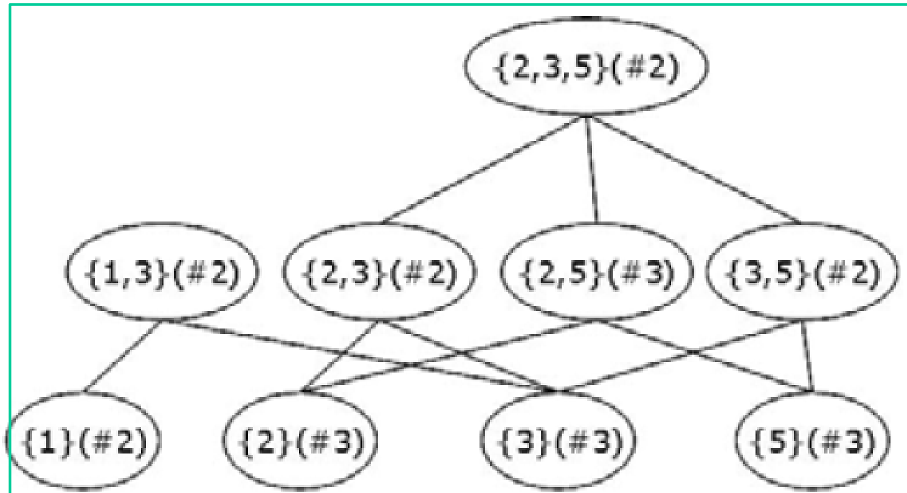


FI

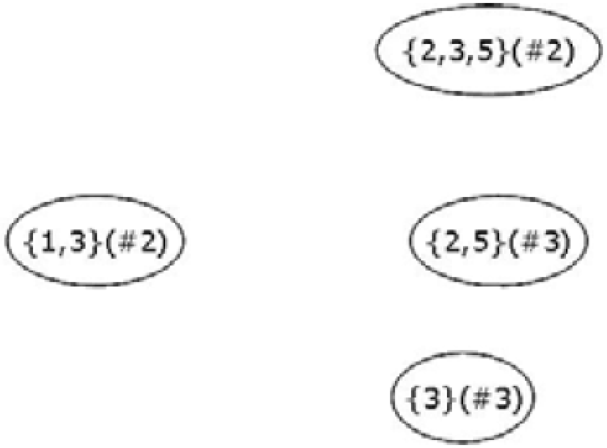


MFI

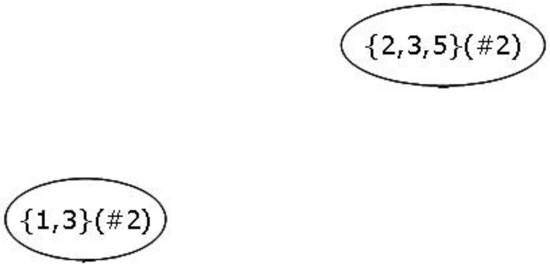
FIs – CFIs - MFIs



FI



CFI



MFI

Outline

- Introduction
- Basic concepts
- Frequent Itemsets Mining (FIM) – Apriori
- Association Rules Mining
- Apriori improvements
- Closed frequent itemsets (CFI) & Maximal frequent itemsets (MFI)
- Things you should know
- Homework/tutorial

Things you should know

- Frequent Itemsets, support, minSupport, itemsets lattice
- Association Rules, support, minSupport, confidence, minConfidence, strong rules
- Frequent Itemsets Mining: computation cost, negative border, downward closure property
- Apriori: join step, prune step, DB scans
- Association rules extraction from frequent itemsets
- Quality measures for association rules
- Improvements of Apriori (Partition, Sampling, FPGrowth, Eclat)
- Horizontal representation / Vertical representation
- Closed Frequent Itemsets (CFI)
- Maximal Frequent Itemsets (MFI)

Homework/ Tutorial

Tutorial: 2nd tutorial on Thursday on:

- Frequent Itemsets and Association rules
- Detecting frequent itemsets/ association rules in a real dataset from stack overflow (http://stackoverflow.com/).

Homework: Have a look at the tutorial in the website and try to solve the exercises.

- Try to run Apriori in Weka using dataset weather.nominal.arff (in weka installation folder/data)
  - A bigger dataset supermarket.arff (in weka installation folder/data)
- Try to implement Apriori ☺

Suggested reading:

- Han J., KamberM., Pei J. *Data Mining: Concepts and Techniques 3rd ed., Morgan Kaufmann, 2011* (Chapter 6)
- Apriori algorithm: Rakesh Agrawal and R. Srikant, *Fast Algorithms for Mining Association Rules*, VLDB'94.