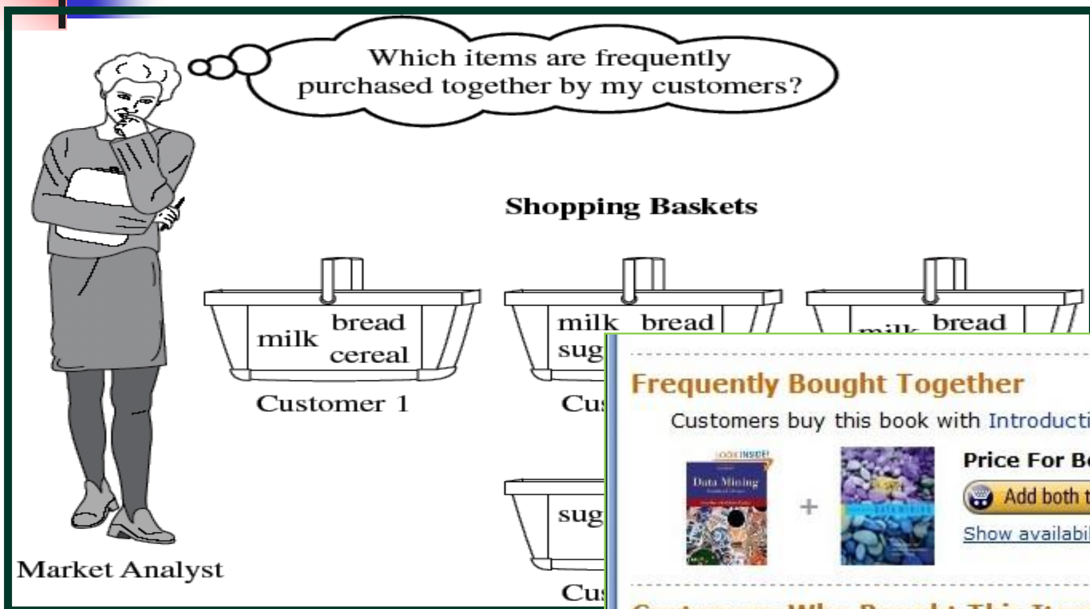
Lecture 10 - Discovery of Association Rules

- Basic Concepts
- The Apriori Algorithm for mining of (single-dimensional Boolean) association rules in transactional databases
- Visualization of Association Rules
- Improving the Efficiency of Apriori

Motivating Examples



Market Basket Analysis

amazon.com

What Is Association Rule Mining?

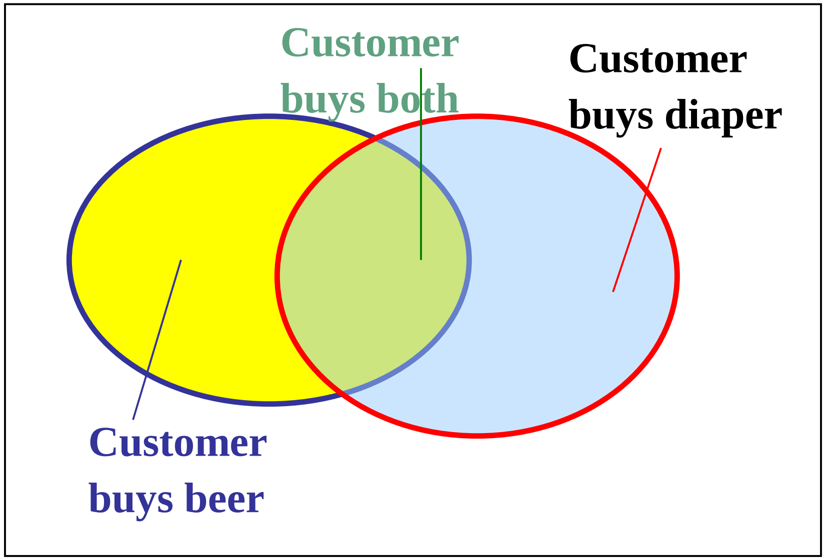
- Motivation: finding regularities in data
  - What products were often purchased together? — Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?
- Association rule mining:
  - Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.
  - Frequent pattern: pattern (set of items, sequence, etc.) that occurs frequently in a database [AIS93]

Why Is Frequent Pattern or Association Mining an Essential Task in Data Mining?

- Foundation for many essential data mining tasks
  - Association, correlation, causality
  - Sequential patterns, temporal or cyclic association, partial periodicity, spatial and multimedia association
  - Associative classification, cluster analysis, iceberg cube, fascicles (semantic data compression based on similar attribute values)
- Broad applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis
  - Web log (click stream) analysis, DNA sequence analysis, etc.

Basic Concepts: Frequent Patterns and Association Rules

Transaction-id	Items bought
10	A, B, C
20	A, C
30	A, D
40	B, E, F



- Itemset $T = \{x_1, \ldots, x_k\}$
- $X \subset T, Y \subset T$
- Find all the rules $X \rightarrow Y$ with min confidence and support
  - Support $(X \rightarrow Y)$, *s*, probability that a transaction contains the *union* of *X* and *Y*
    - $P(X \cup Y)$
  - Confidence $(X \rightarrow Y)$, *c*, conditional probability that a transaction having X also contains *Y*
    - $P(Y / X)$

Mining Association Rules—an Example

Transaction-id	Items bought
10	A, B, C
20	A, C
30	A, D
40	B, E, F

Min. support 50%
Min. confidence 50%

Frequent pattern	Support
{A}	75%
{B}	50%
{C}	50%
{A, C}	50%

For rule $A \Rightarrow C$:

support = support($\{A\} \cup \{C\}$) = 50%

confidence = support($\{A\} \cup \{C\}$)/support($\{A\}$) = 66.6%

Mining Association Rules – Main Steps

- Find all frequent itemsets
  - Each itemset will occur in at least *min_sup* database transactions
- Generate **strong** association rules from the frequent itemsets
  - These rules must satisfy *min_sup* and *min_conf*

Closed Patterns and Max-Patterns

- A long pattern contains a combinatorial number of sub-patterns, e.g., $\{a_1, \ldots, a_{100}\}$ contains $\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100} - 1 \approx 1.27 \times 10^{30}$ sub-patterns!
  - Amazon.com: 1,800,000 book titles
- Solution: *Mine closed patterns and max-patterns instead*
- An itemset X is closed if X is *frequent* and there exists *no super-pattern* $Y \supset X$, *with the same support* as X
- An itemset X is a max-pattern if X is frequent and there exists no frequent super-pattern $Y \supset X$
- Closed pattern is a lossless compression of freq. patterns
  - Reducing the # of patterns and rules

Closed Patterns and Max-Patterns

- Exercise. DB = {<a_1, …, a_{100}>, < a_1, …, a_{50}>}
  - Min_sup = 1.
- What is the set of closed itemsets?
  - <a_1, …, a_{100}>: 1
  - < a_1, …, a_{50}>: 2
- What is the set of max-pattern?
  - <a_1, …, a_{100}>: 1
- What is the set of all patterns?
  - !!

Computational Complexity of Frequent Itemset Mining

- How many itemsets are potentially to be generated in the worst case?
  - The number of frequent itemsets to be generated is sensitive to the minsup threshold
  - When minsup is low, there exist potentially an exponential number of frequent itemsets
  - The worst case: M^N where M: # distinct items, and N: max length of transactions
- The worst case complexty vs. the expected probability
  - Ex. Suppose Walmart has 10^4 kinds of products
    - The chance to pick up one product 10^{-4}
    - The chance to pick up a particular set of 10 products: $\sim 10^{-40}$
    - What is the chance this particular set of 10 products to be frequent 10^3 times in 10^9 transactions?

Lecture 10 - Discovery of Association Rules

- Basic Concepts
- The Apriori Algorithm for mining of (single-dimensional Boolean) association rules in transactional databases
- Visualization of Association Rules
- Improving the Efficiency of Apriori

Apriori: A Candidate Generation-and-test Approach

- Any subset of a frequent itemset must be frequent
  - if **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}**
  - Every transaction having {beer, diaper, nuts} also contains {beer, diaper}
- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested!
- Method:
  - Generate length (k+1) candidate itemsets from length k frequent itemsets by joining them with themselves
  - Prune k-itemsets containing infrequent (k-1)-itemsets
  - Test the remaining candidates against DB
- The performance studies show its efficiency and scalability

Min. support 50%

The Apriori Algorithm — An Example

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

The Apriori Algorithm Outline

- Pseudo-code:

  C_k: Candidate itemset of size k
  L_k : frequent itemset of size k
  Input: *D*, a database of transactions; *min_sup*
  Output: *L*, frequent itemsets in *D*

  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 L = $\cup_k L_k$;

Important Details of Apriori

- How to generate candidates?
  - Step 1: self-joining L_k
  - Step 2: pruning
- How to count supports of candidates?
- Example of Candidate-generation
  - L_3={*abc, abd, acd, ace, bcd*}
  - Self-joining: $L_3 * L_3$
    - *abcd* from *abc* and *abd*
    - *acde* from *acd* and *ace*
  - Pruning:
    - *acde* is removed because *ade* is not in L_3
  - C_4={*abcd*}

How to Generate Candidates?

- Suppose the items in L_{k-1} are listed in a lexicographic order
- Step 1: self-joining L_{k-1}

  insert into C_k

  select $p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}$

  from L_{k-1} p, L_{k-1} q

  where $p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$ //*First k* - 2 items are identical
- Step 2: pruning

  forall ***itemsets c in C_k*** do

  forall ***(k-1)-subsets s of c*** do

  if *(s is not in L_{k-1})* **then delete** *c* **from** C_k

How to Count Supports of Candidates?

- Why counting supports of candidates a problem?
  - The total number of candidates can be very huge
  - One transaction may contain many candidates
- Method:
  - Candidate itemsets are stored in a *hash-tree*
  - *Leaf* node of hash-tree contains a list of itemsets and counts
  - *Interior* node contains a hash table
  - *Subset function*: finds all the candidates contained in a transaction

Further Improvement of the Apriori Method

- Major computational challenges
  - Multiple scans of transaction database
  - Huge number of candidates
  - Tedious workload of support counting for candidates
- Improving Apriori: general ideas
  - Reduce passes of transaction database scans
  - Shrink number of candidates
  - Facilitate support counting of candidates

Interestingness Measure: Correlations (Lift)

- *play basketball* $\Rightarrow$ *eat cereal* [40%, 66.7%] is misleading – why?
  - The overall % of students eating cereal is 75% > 66.7%.
- *play basketball* $\Rightarrow$ *not eat cereal* [20%, 33.3%] is more accurate, although with lower support and confidence
- Measure of dependent/correlated events: lift

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

	Basketball	Not basketball	Sum (row)
Cereal	2000	1750	3750
Not cereal	1000	250	1250
Sum(col.)	3000	2000	5000

$$lift(B,C) = \frac{2000/5000}{3000/5000 * 3750/5000} = 0.89 \qquad lift(B,\neg C) = \frac{1000/5000}{3000/5000 * 1250/5000} = 1.33$$

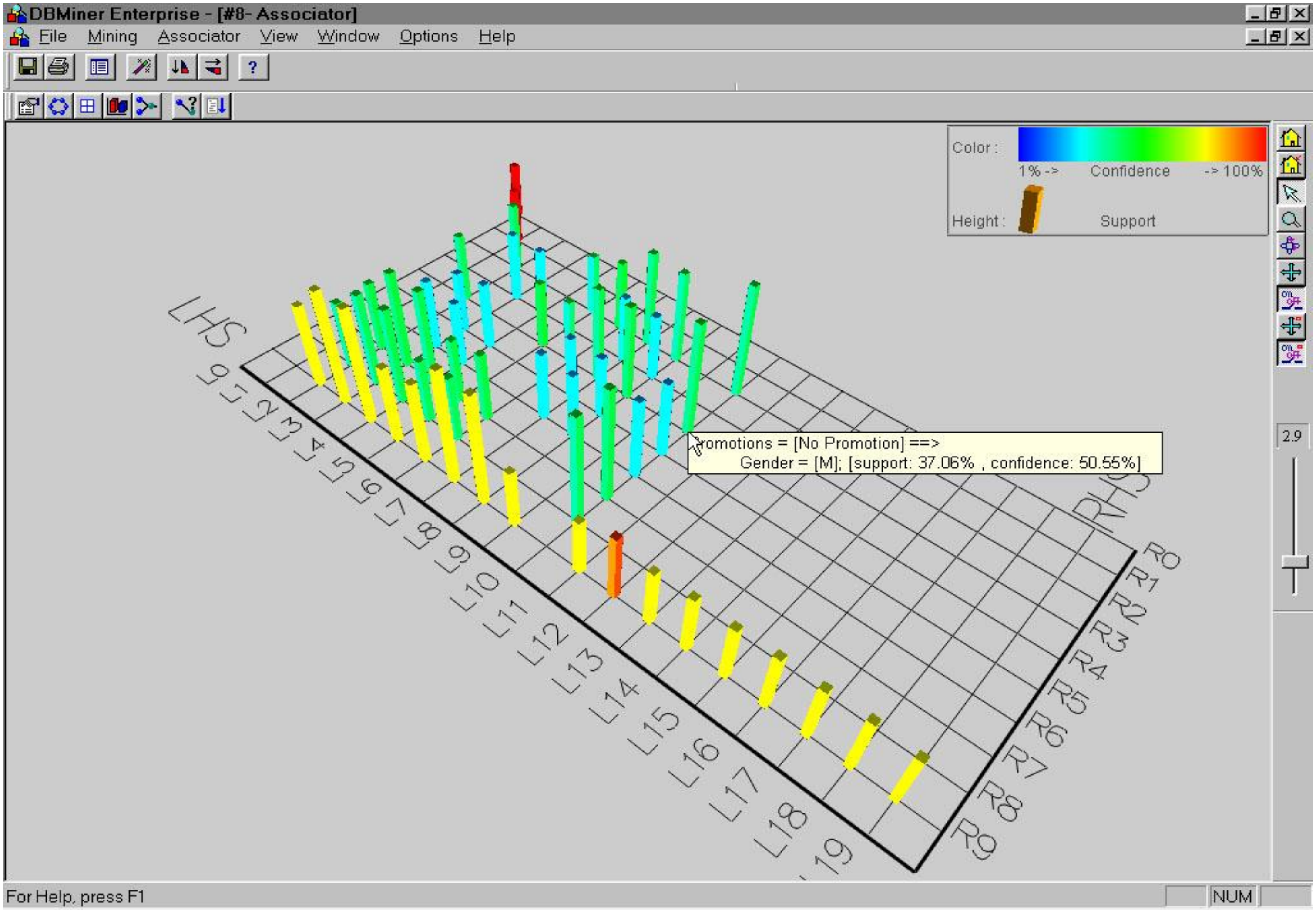
Mining Association Rules in Temporal Databases

- *Episode rule* - applied to sequences of events, each event occurring at a particular time. An *episode* is a sequence of events in a partial order and an *episode rule* is defined as $A \rightarrow B$, where *A* and *B* are *episodes* and *A* is a *subepisode* of *B*. Can be used for predicting certain events by sequences of earlier events ("if the *Microsoft* stock price goes up, then *IBM* goes up the next day")
- *Trend dependencies* - a rule $A \rightarrow B$, where *A* and *B* are patterns of the form (A, Θ), where A is a reference to a specific attribute and Θ is an element in $\{<,=,>,\geq, \leq, =\}$. Can be used to discover patterns like: *an employee's salary always increases over time*.
- *Sequence rules* - $\{AB, A\} \rightarrow \{C, A\}$, where *A* and *B* appears at the same time followed by *A*, implies a sequence where *C* is followed by *A*.
- *Calendric rules* - predefined time unit and a calendar, given by a set of time intervals, are needed. Rules $A \rightarrow B$ restricted to the chosen calendar, which allows for seasonal changes to be analyzed. Example: sales are up before Xmas.
- *Intertransaction rules, Interval AR, etc.*

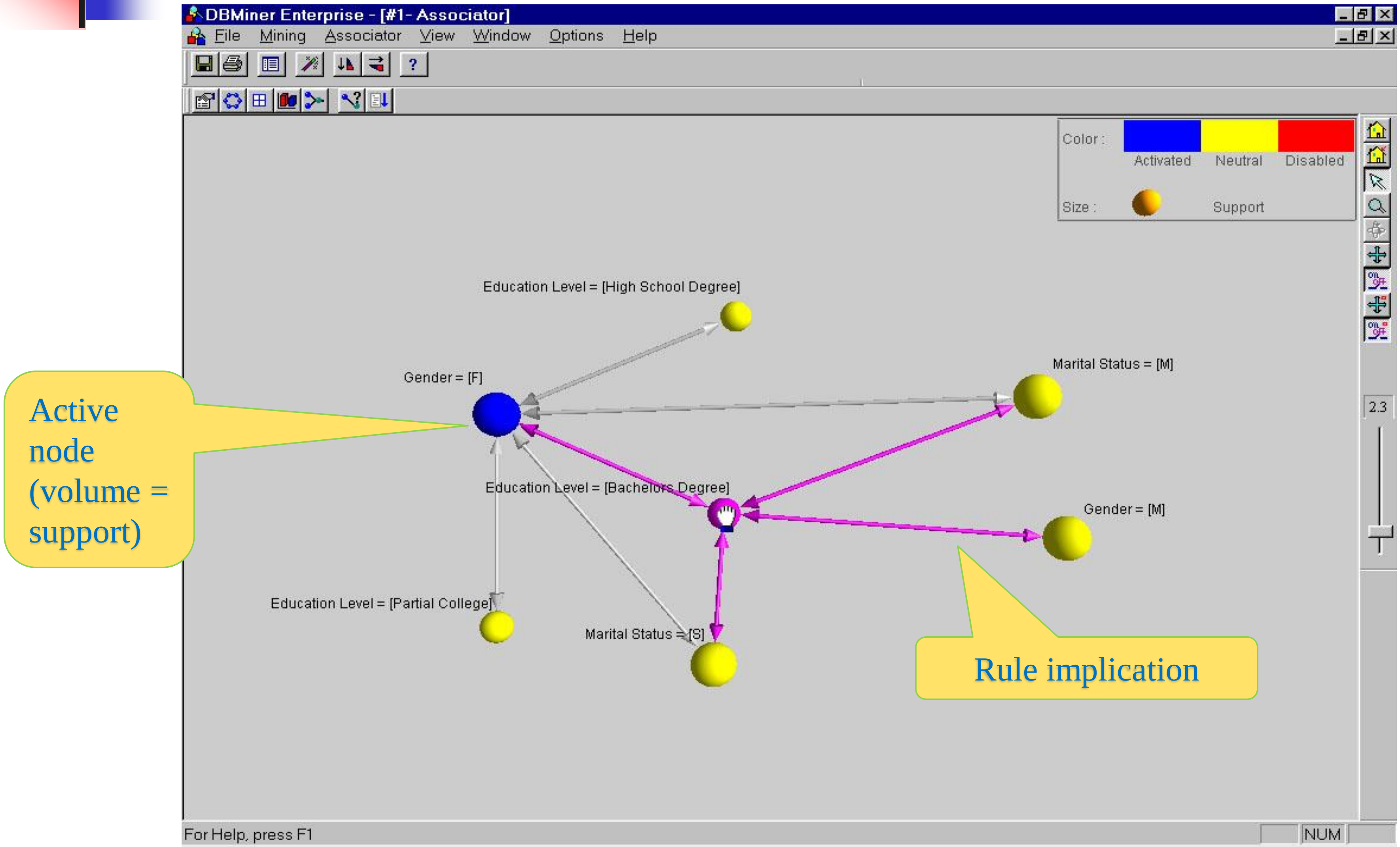
Lecture 10 - Discovery of Association Rules

- Basic Concepts
- The Apriori Algorithm for mining of (single-dimensional Boolean) association rules in transactional databases
- Visualization of Association Rules
- Improving the Efficiency of Apriori

Visualization of Association Rules: Plane Graph



Visualization of Association Rules: A Ball Graph



Lesson 9 - Discovery of Association Rules

- Basic Concepts
- The Apriori Algorithm for mining of (single-dimensional Boolean) association rules in transactional databases
- Visualization of Association Rules
- Improving the Efficiency of Apriori

Challenges of Frequent Pattern Mining

- Challenges
  - Multiple scans of transaction database
  - Huge number of candidates
  - Tedious workload of support counting for candidates
- Improving Apriori: general ideas
  - Reduce passes of transaction database scans
  - Shrink number of candidates
  - Facilitate support counting of candidates

Partition: Scan Database Only Twice

- Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
  - Scan 1: partition database and find local frequent patterns
  - Scan 2: consolidate global frequent patterns
- A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association in large databases. In *VLDB'95*
  - *www: http://www.acm.org/sigmod/vldb/conf/1995/P432.PDF*

Bottleneck of Frequent-pattern Mining

- Multiple database scans are costly
- Mining long patterns needs many passes of scanning and generates lots of candidates
  - To find frequent itemset $i_1 i_2 \ldots i_{100}$
    - # of scans: 100
    - # of Candidates: $\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100}-1 \approx 1.27*10^{30}$!
- Bottleneck: candidate-generation-and-test
- Can we avoid candidate generation?

Mining Frequent Patterns Without Candidate Generation

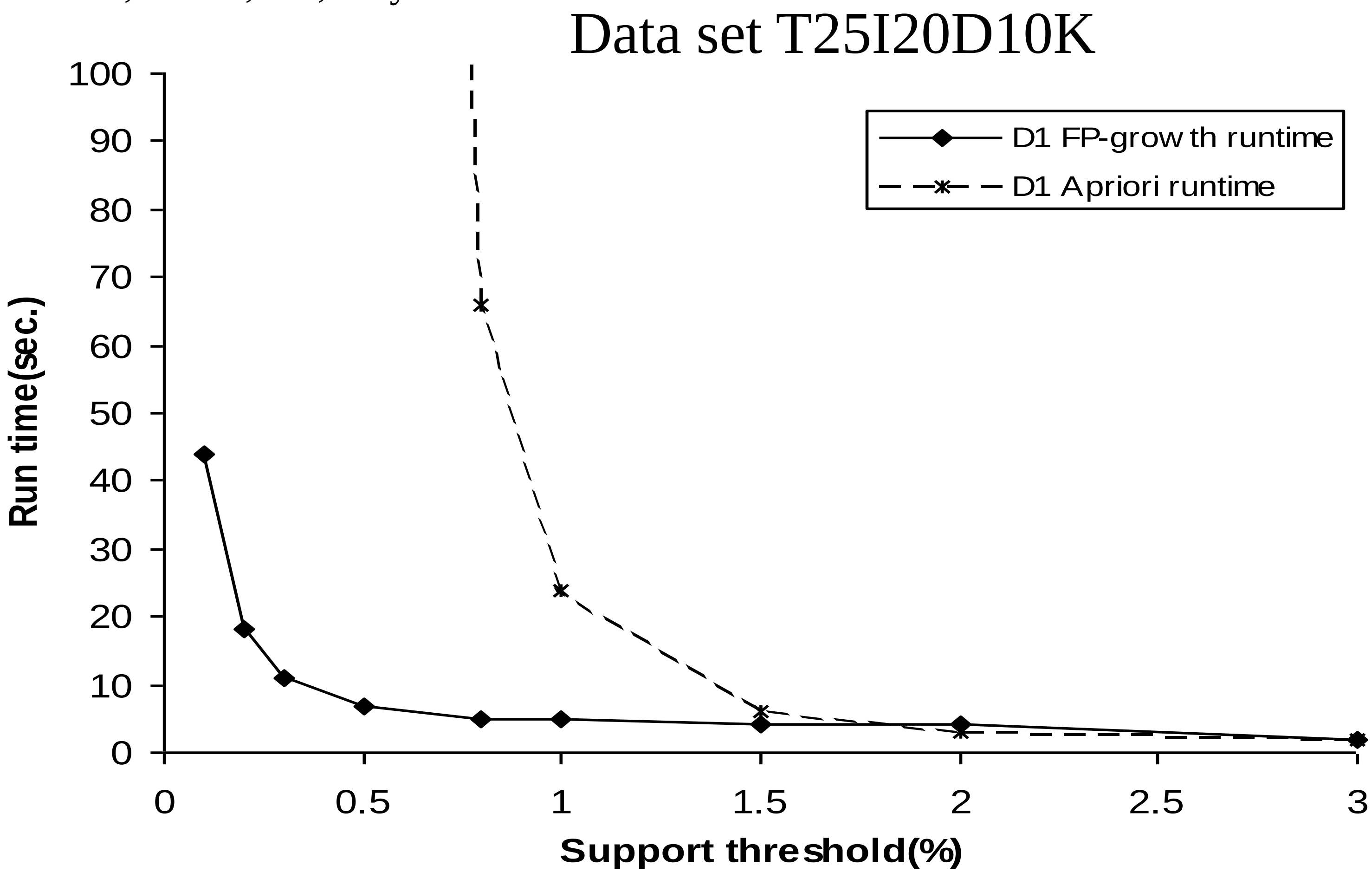
- **Grow long patterns from short ones using local frequent items**
  - "abc" is a frequent pattern
  - Get all transactions having "abc": DB|abc
  - "d" is a local frequent item in DB|abc → abcd is a potentially frequent pattern
- **"Divide-and-conquer" strategy**
  - Compress the database into a frequent-pattern (FP) tree
  - Divide a compressed database into a set of *conditional databases*, each associated with one frequent item
  - Mine each database separately

Mining Frequent Patterns With FP-trees

- Idea: Frequent pattern growth
  - Recursively grow frequent patterns by pattern and database partition
- Method
  - For each frequent item, construct its conditional pattern-base, and then its conditional FP-tree
  - Repeat the process on each newly created conditional FP-tree
  - Until the resulting FP-tree is empty, or it contains only one path—single path will generate all the combinations of its sub-paths, each of which is a frequent pattern

FP-Growth vs. Apriori: Scalability With the Support Threshold

Source: J. Han, J. Pei, and Y. Yin. Mining Frequent Patterns without Candidate Generation. In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, TX, May 2000.



Why Is FP-Growth the Winner?

- Divide-and-conquer:
  - decompose both the mining task and DB according to the frequent patterns obtained so far
  - leads to focused search of smaller databases
- Other factors
  - no candidate generation, no candidate test
  - compressed database: FP-tree structure
  - no repeated scan of entire database
  - basic ops—counting local freq items and building sub FP-tree, no pattern search and matching

Summary

- Frequent pattern mining—an important task in data mining
- Scalable frequent pattern mining methods
  - Apriori (Candidate generation & test)
  - Projection-based (FPgrowth, CLOSET+, ...)
  - Vertical format approach (CHARM, ...)
- Which patterns are interesting?
  - Pattern evaluation methods