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
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 number of transaction database scans
  - Shrink number of candidates
  - Facilitate support counting of candidates

DIC: Reduce Number of Scans

Once both A and D are determined frequent, the counting of AD can begin.

Once all length-2 subsets of BCD are determined frequent, the counting of BCD can begin.

Transactions

1-itemsets

2-itemsets

... 

1-itemsets

2-items

3-items

DHP: Reduce the Number of Candidates

- A hashing bucket count $\text{min}_\text{sup} \rightarrow$ every candidate in the bucket is infrequent
  - Candidates: a, b, c, d, e
  - Hash entries: \{ab, ad, ae\} \{bd, be, de\} …
  - Large 1-itemset: a, b, d, e
  - The sum of counts of \{ab, ad, ae\} $< \text{min}_\text{sup} \rightarrow$ ab should not be a candidate 2-itemset
- J. Park, M. Chen, and P. Yu, 1995

Partition: Scan Database Only Twice

- Partition the database into n partitions
- Itemset $X$ is frequent $\rightarrow$ $X$ is frequent in at least one partition
  - Scan 1: partition database and find local frequent patterns
  - Scan 2: consolidate global frequent patterns
- A. Savasere, E. Omiecinski, and S. Navathe, 1995
Sampling for Frequent Patterns

- Select a sample of original database, mine frequent patterns within sample using Apriori
- Scan database once to verify frequent itemsets found in sample, only borders of closure of frequent patterns are checked
  - Example: check abcd instead of ab, ac, ..., etc.
- Scan database again to find missed frequent patterns
- H. Toivonen, 1996

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 $i_1i_2...i_{100}$
    - # of scans: 100
    - # of Candidates: $\binom{100}{1} + \binom{100}{2} + \cdots + \binom{100}{100} = 2^{100} - 1 \approx 1.27 \times 10^{30}$
  - Bottleneck: candidate-generation-and-test
- Can we avoid candidate generation?
Set Enumeration Tree

- Subsets of $I$ can be enumerated systematically
  - $I=\{a, b, c, d\}$

Borders of Frequent Itemsets

- Connected
  - X and Y are frequent and X is an ancestor of Y
    - all patterns between X and Y are frequent
Projected Databases

- To find a child $X_y$ of $X$, only $X$-projected database is needed
  - The sub-database of transactions containing $X$
  - Item $y$ is frequent in $X$-projected database

![Diagram showing tree projection]

Tree-Projection Method

- Find frequent 2-itemsets
- For each frequent 2-itemset $xy$, form a projected database
  - The sub-database containing $xy$
- Recursive mining
  - If $x'y'$ is frequent in $xy$-proj db, then $xyx'y'$ is a frequent pattern
Compress Database by FP-tree

- 1st scan: find freq items
  - Only record freq items in FP-tree
  - F-list: f-c-a-b-m-p
- 2nd scan: construct tree
  - Order freq items in each transaction w.r.t. f-list
  - Explore sharing among transactions

Benefits of FP-tree

- Completeness
  - Never break a long pattern in any transaction
  - Preserve complete information for freq pattern mining
    - No need to scan database anymore
- Compactness
  - Reduce irrelevant info — infrequent items are gone
  - Items in frequency descending order (f-list): the more frequently occurring, the more likely to be shared
  - Never be larger than the original database (not counting node-links and the count fields)
Partition Frequent Patterns

- Frequent patterns can be partitioned into subsets according to 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, or p
  - Pattern f
- The partitioning is complete and without any overlap

Find Patterns Having Item “p”

- Only transactions containing p are needed
- Form p-projected database
  - Starting at entry p of header table
  - Follow the side-link of frequent item p
  - Accumulate all transformed prefix paths of p

p-projected database $TDB|_p$

- Local frequent item: c:3
- Frequent patterns containing p
  - p: 3, pc: 3
Find Patterns Having Item m But No p

- Form m-projected database TDB|m
  - Item p is excluded
  - Contain fca:2, fcab:1
  - Local frequent items: f, c, a
- Build FP-tree for TDB|m

Recursive Mining

- Patterns having m but no p can be mined recursively
- Optimization: enumerate patterns from single-branch FP-tree
  - Enumerate all combination
  - Support = that of the last item
    - m, fm, cm, am
    - fcm, fam, cam
    - fcm
Borders and Max-patterns

- Max-patterns: borders of frequent patterns
  - A subset of max-pattern is frequent
  - A superset of max-pattern is infrequent

MaxMiner: Mining Max-patterns

- 1st scan: find frequent items
  - A, B, C, D, E
- 2nd scan: find support for
  - AB, AC, AD, AE, ABCDE
  - BC, BD, BE, BCDE
  - CD, CE, CDE, DE
- Since BCDE is a max-pattern, no need to check BCD, BDE, CDE in later scan
- Baya’98

<table>
<thead>
<tr>
<th>Tid</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>A,B,C,D,E</td>
</tr>
<tr>
<td>20</td>
<td>B,C,D,E,</td>
</tr>
<tr>
<td>30</td>
<td>A,C,D,F</td>
</tr>
<tr>
<td></td>
<td>Min_sup=2</td>
</tr>
<tr>
<td></td>
<td>Potential</td>
</tr>
<tr>
<td></td>
<td>max-patterns</td>
</tr>
</tbody>
</table>

COMP 790-090 Data Mining: Concepts, Algorithms, and Applications
Frequent Closed Patterns

- For frequent itemset $X$, if there exists no item $y$ s.t. every transaction containing $X$ also contains $y$, then $X$ is a frequent closed pattern.
  - “acdf” is a frequent closed pattern
- Concise rep. of freq pats
- Reduce # of patterns and rules
- N. Pasquier et al. In ICDT’99

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>a, c, d, e, f</td>
</tr>
<tr>
<td>20</td>
<td>a, b, e</td>
</tr>
<tr>
<td>30</td>
<td>c, e, f</td>
</tr>
<tr>
<td>40</td>
<td>a, c, d, f</td>
</tr>
<tr>
<td>50</td>
<td>c, e, f</td>
</tr>
</tbody>
</table>

Min_sup=2

CLOSESET: Mining Frequent Closed Patterns

- Flist: list of all freq items in support asc. order
  - Flist: d-a-f-e-c
- Divide search space
  - Patterns having $d$
  - Patterns having $d$ but no $a$, etc.
- Find frequent closed pattern recursively
  - Every transaction having $d$ also has $cfa \Rightarrow$ cfad is a frequent closed pattern
- PHM’00
Closed and Max-patterns

- Closed pattern mining algorithms can be adapted to mine max-patterns
  - A max-pattern must be closed
- Depth-first search methods have advantages over breadth-first search ones

Mining Various Kinds of Rules or Regularities

- Multi-level, quantitative association rules, correlation and causality, ratio rules, sequential patterns, emerging patterns, temporal associations, partial periodicity
- Classification, clustering, iceberg cubes, etc.
Multiple-level Association Rules

- Items often form hierarchy
- Flexible support settings: Items at the lower level are expected to have lower support.
- Transaction database can be encoded based on dimensions and levels
- Explore shared multi-level mining

<table>
<thead>
<tr>
<th>uniform support</th>
<th>reduced support</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Level 1</strong></td>
<td><strong>Level 1</strong></td>
</tr>
<tr>
<td>min_sup = 5%</td>
<td>min_sup = 5%</td>
</tr>
<tr>
<td>![Milk](support = 10%)</td>
<td>![Milk](support = 10%)</td>
</tr>
<tr>
<td><strong>Level 2</strong></td>
<td><strong>Level 2</strong></td>
</tr>
<tr>
<td>min_sup = 5%</td>
<td>min_sup = 3%</td>
</tr>
<tr>
<td>![2% Milk](support = 6%)</td>
<td>![2% Milk](support = 6%)</td>
</tr>
<tr>
<td>![Skim Milk](support = 4%)</td>
<td>![Skim Milk](support = 4%)</td>
</tr>
</tbody>
</table>

---

Multi-dimensional Association Rules

- Single-dimensional rules:
  - buys(X, “milk”) ⇒ buys(X, “bread”)
- MD rules: ≥ 2 dimensions or predicates
  - Inter-dimension assoc. rules (no repeated predicates)
    - age(X,”19-25”) ∧ occupation(X,“student”) ⇒ buys(X,“coke”)
  - hybrid-dimension assoc. rules (repeated predicates)
    - age(X,”19-25”) ∧ buys(X, “popcorn”) ⇒ buys(X, “coke”)
- Categorical Attributes: finite number of possible values, no order among values
- Quantitative Attributes: numeric, implicit order
Quantitative/Weighted Association Rules

Numeric attributes are *dynamically* discretized to maximize the confidence or compactness of the rules.

2-D quantitative association rules: $A_{\text{quan}1} \land A_{\text{quan}2} \Rightarrow A_{\text{cat}}$

Cluster “adjacent” association rules to form general rules using a 2-D grid.

**age(X,”33-34”) \land income(X,"30K - 50K”) \Rightarrow buys(X,”high resolution TV”)**

---

**Mining Distance-based Association Rules**

- Binning methods do not capture semantics of interval data
- Distance-based partitioning
  - Density/number of points in an interval
  - “Closeness” of points in an interval

<table>
<thead>
<tr>
<th>Price</th>
<th>Equi-width</th>
<th>Equi-depth</th>
<th>Distance-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>[0,10]</td>
<td>[7,20]</td>
<td>[7,7]</td>
</tr>
<tr>
<td>20</td>
<td>[11,20]</td>
<td>[22,50]</td>
<td>[20,22]</td>
</tr>
<tr>
<td>22</td>
<td>[21,30]</td>
<td>[51,53]</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>[31,40]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>[41,50]</td>
<td>[50,53]</td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>[51,60]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Constraint-based Data Mining

- Find all the patterns in a database autonomously?
  - The patterns could be too many but not focused!
- Data mining should be interactive
  - User directs what to be mined
- Constraint-based mining
  - User flexibility: provides constraints on what to be mined
  - System optimization: push constraints for efficient mining

Constraints in Data Mining

- Knowledge type constraint
  - classification, association, etc.
- Data constraint — using SQL-like queries
  - find product pairs sold together in stores in New York
- Dimension/level constraint
  - in relevance to region, price, brand, customer category
- Rule (or pattern) constraint
  - small sales (price < $10) triggers big sales (sum >$200)
- Interestingness constraint
  - strong rules: support and confidence