Incremental Mining of Frequent Itemsets

CPS 296.1

Topics in Database Systems

Mining a growing database

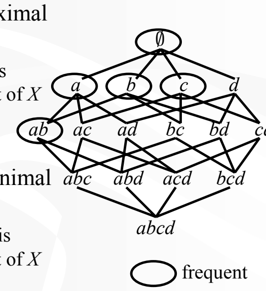
- Given: *DB*, a database of transactions, each containing a set of items
- Find: $L(DB)$, the set of all frequent itemsets
  - A set of items X is frequent if no less than $s_{min}\% \times |DB|$ transactions contain X
- If we add a set of transaction to the database (i.e., $DB \leftarrow DB \uplus \triangle DB$), what is $L(DB \uplus \triangle DB)$?
  - Re-computation is not optimal because it ignores the result of mining the old *DB*

Incorrect approaches

- $L(DB \uplus \triangle DB) = L(DB) \cup L(\triangle DB)$?
  - X can be frequent in *DB*, but it can be infrequent in $\triangle DB$ and $DB \uplus \triangle DB$
  - And vice versa: X can be frequent in $\triangle DB$, but it can be infrequent in *DB* and $DB \uplus \triangle DB$
  - ➢ $L(DB)$ is not monotone
- $L(DB \uplus \triangle DB) = L(DB) \cap L(\triangle DB)$?
  - X can be infrequent in *DB*, but it can be frequent in $\triangle DB$ and $DB \uplus \triangle DB$
  - And vice versa

Positive and negative border

- Positive border, $bd^+(DB)$: maximal frequent itemsets in *DB*
  - X is in the positive border if X is frequent and no proper superset of X is frequent
  - Example: $bd^+(DB) = \{ ab, c \}$
- Negative border, $bd^-(DB)$: minimal infrequent itemsets in *DB*
  - X is in the negative border if X is infrequent and no proper subset of X is infrequent
  - Example: $bd^-(DB) = \{ d, ac, bc \}$



Facts about negative border

- Observation 1: Every 1-itemset is in either $L(DB)$ or $bd^-(DB)$
- Observation 2: recall pass k of Apriori
  - Generate C_k (candidate itemsets of size k) from L_{k-1} (frequent itemsets of size $k-1$)
  - Count C_k to determine L_k ($\subseteq C_k$)
  - ➢ $C_k - L_k$ is the negative border at level k
  - ➢ Apriori counts $C_k - L_k$
- ➢ After mining *DB*, we know itemsets in both $L(DB)$ and $bd^-(DB)$, together with their counts
  - Remember such information to help the incremental mining algorithm

First try at an incremental algorithm

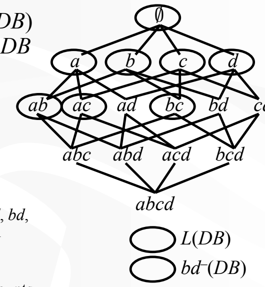
- Input: DB, $\triangle DB$, $L(DB)$ and $bd^-(DB)$ together with their counts in *DB*
- Output: $L(DB \uplus \triangle DB)$ together with their counts in $DB \uplus \triangle DB$ (← will come back later to this requirement)
- Method
  - Same as Apriori, but
  - When counting C_k, if $X \in C_k$ is in $L(DB)$ or $bd^-(DB)$, do not go through *DB* because the count of X in *DB* is already known; simply go through $\triangle DB$
  - ➢ We might save a scan over *DB* (but not $\triangle DB$) if all itemsets in C_k have been counted in *DB*

A problem of the first try

- Each scan over *DB* may count only a few itemsets → insufficient computation to overlap I/O
  - Also a problem in Apriori
  - But aggravated in the incremental algorithm because some of C_k may have been counted before
- ➢ In general, a trade-off in level-wise algorithms
  - If we count an itemset X in the next level, we risk doing useless work because a subset of X (which we are counting at the same time) may turn out to be infrequent

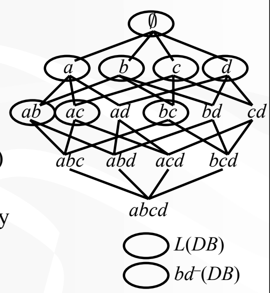
Another problem (?) (slide 1)

- Did not use the fact that $bd^-(DB)$ and beyond are infrequent in *DB*
- Example
  - Pass 1
    - Scan of *DB* is saved
    - Say all 1-itemsets turn out to be frequent in $DB \uplus \triangle DB$
  - Pass 2
    - Scan of *DB* is needed because *ad*, *bd*, $cd \in C_2$ but they have never been counted in *DB*
    - But at least we know they are infrequent in *DB*; perhaps their counts in $\triangle DB$ are not high enough to make them frequent in $DB \uplus \triangle DB$, so we could have avoided scanning *DB*



Another problem (?) (slide 2)

- If we also care about $bd^-(DB \uplus \triangle DB)$ with counts, then we still need to count *ad*, *bd*, *cd* in *DB*
  - Counts for $bd^-(DB \uplus \triangle DB)$ are needed to make the incremental algorithm ready for next $\triangle DB$



Observation 1

- If X is infrequent in *DB*, then X can be frequent in $DB \uplus \triangle DB$ only if X is frequent in $\triangle DB$
  - Infrequent in both *DB* and $\triangle DB$ → infrequent in $DB \uplus \triangle DB$
- ➢ Strategy implied
  - First, mine $\triangle DB$ to find $L(\triangle DB)$ and $bd^-(\triangle DB)$ with counts
  - When counting C_k, if $X \in C_k$ is in $L(\triangle DB)$ or $bd^-(\triangle DB)$, do not go through $\triangle DB$ because the count of X in $\triangle DB$ is already known
  - Add the following pruning condition: For any $X \in C_k$, if we already know $X \notin L(DB)$ and $X \notin L(\triangle DB)$, remove X from C_k

Observation 2

- If none of the itemsets in $bd^-(DB)$ becomes frequent in $DB \uplus \triangle DB$, then no new itemset will be introduced (i.e., $L(DB \uplus \triangle DB) \subseteq L(DB)$)
  - Say X is infrequent in *DB*
  - Then there exists $Y \subseteq X$ s.t. $Y \in bd^-(DB)$
  - Since none of the itemsets in $bd^-(DB)$ is frequent in $DB \uplus \triangle DB$, Y is infrequent in $DB \uplus \triangle DB$
  - That means $X \supseteq Y$ is infrequent in $DB \uplus \triangle DB$
- ➢ Strategy implied
  - In $\triangle DB$, count itemsets in $bd^-(DB)$ to find their counts in $DB \uplus \triangle DB$
  - If none of these itemsets are frequent in $DB \uplus \triangle DB$, there is no need to scan *DB* at all

Second try (slide 1)

➢ Thomas et al. "An Efficient Algorithm for the Incremental Updation of Association Rules in Large Databases." *SIGKDD*, 1997

- Mine $\triangle DB$ to obtain $L(\triangle DB)$ and $bd^-(\triangle DB)$ with counts
- While mining $\triangle DB$, also count itemsets in $L(DB)$ and $bd^-(DB)$
- For each itemset in $L(DB)$ and $bd^-(DB)$, calculate its count in $DB \uplus \triangle DB$

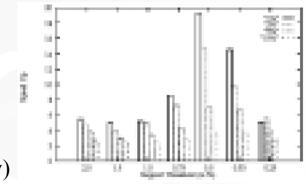
(Continue on the next slide)

Second try (slide 2)

(Continued from the previous slide)

- If none of the itemsets in $bd^-(DB)$ is frequent in $DB \uplus \triangle DB$, stop and output itemsets in $L(DB)$ and $bd^-(DB)$ that are in $L(DB \uplus \triangle DB)$ or $bd^-(DB \uplus \triangle DB)$, together with their counts
- Otherwise, scan DB once
  - Count all itemsets in $C =$
    $L(\triangle DB) \cup bd^-(\triangle DB) - L(DB) - bd^-(DB) -$
    $\{ X \mid \exists Y \in L(DB) \cup bd^-(DB)$ s.t. Y is known to be infrequent in $DB \uplus \triangle DB$ and $Y \subseteq X \}$
  - Output itemsets in $L(DB)$, $bd^-(DB)$, and C that are in $L(DB \uplus \triangle DB)$ or $bd^-(DB \uplus \triangle DB)$, together with their counts

Experiments



- Not nearly close to the ideal speed-up
  - Incremental algorithm does not replace Apriori
- Smaller $\triangle DB$ means bigger speed-up (usually)
- Speed-up is lower for very high support threshold
  - Apriori makes very few passes anyway
- Speed-up is lower for very low support threshold
  - Probability of the negative border expanding is higher

First vs. second try

- Second try (Thomas et al.) scans DB at most once
  - May need to count lots of itemsets in the same pass
  - Some of these itemset may not need to be counted
    - Example?
  - Also, complete mining of $\triangle DB$ may be unnecessary
    - Example?
- First try scans DB multiple times (up to the number of scans required by Apriori minus one)
  - Will not scan DB if the second try does not
  - May count very few itemsets in one pass
  - Every itemset counted is necessary

➢ Fundamental trade-off in play again!