SS-ZG548: ADVANCED DATA MINING

04 Incremental Mining

Dr. Kamlesh Tiwari
Assistant Professor, Department of CSIS,
BITS Pilani, Pilani Campus, Rajasthan-333031 INDIA

Aug 21, 2021 ONLINE (WILP @ BITS-Pilani July-Dec 2021)

http://ktiwari.in/adm

Recap: Apriori at work

Association Rule Mining involves the discovery of frequent item-sets based on **support** and **confidence** parameters

Consider transactions T

T_1=(A,B,C)
T_2=(A,F)
T_3=(A,B,C,E)
T_4=(A,B,D,F)
T_5=(C,F)
T_6=(A,B,C)
T_7=(A,B,C,E)
T_8=(C,D,E)
T_9=(B,D,E)

Scan T $\longrightarrow C_1$

Item	Supp
{A}	6
{B}	6
{C}	6
{D}	3
{E}	4
{F}	3

$\longrightarrow L_1$

Item	Supp
{A}	6
{B}	6
{C}	6
{E}	4

C_2

Item
{A,B}
{A,C}
{A,E}
{B,C}
{B,E}
{C,E}

Scan T $\longrightarrow C_2$

Item	Supp
{A,B}	5
{A,C}	4
{A,E}	2
{B,C}	4
{B,E}	3
{C,E}	3

$\longrightarrow L_2$

Item	Supp
{A,B}	5
{A,C}	4
{B,C}	4

C_3

Item
{A,B,C}

Scan T $\longrightarrow C_3$

Item	Supp
{A,B,C}	4

$\longrightarrow L_3$

Item	Supp
{A,B,C}	4

Approaches to discover Association Rules involves Apriori, Hash Based (DHP), Partition Based Algorithm

Recap: incremental databases

Real databases could be dynamic. **Incremental** association rule mining is needed as

$$D' = D - \triangle^{-} + \triangle^{+}$$

D

Δ^{-}

$T_1 = (A, B, C)$
$T_2 = (A, F)$
$T_3 = (A, B, C, E)$

$T_4 = (A, B, D, F)$
$T_5 = (C, F)$
$T_6 = (A, B, C)$

D'

$T_7 = (A, B, C, E)$
$T_8 = (C, D, E)$
$T_9 = (B, D, E)$

Δ^{+}

$T_{10} = (B, D)$
$T_{11} = (D, F)$
$T_{12} = (A, B, C, D)$

Recap: FUP[1] can handle insertions

Consider the database D and the related frequent set discovered with Apriori

$T_1 = (A, B, C)$
$T_2 = (A, F)$
$T_3 = (A, B, C, E)$
$T_4 = (A, B, D, F)$
$T_5 = (C, F)$
$T_6 = (A, B, C)$
$T_7 = (A, B, C, E)$
$T_8 = (C, D, E)$
$T_9 = (B, D, E)$

Item set	Support
{A}	6/9
{B}	6/9
{C}	6/9
{E}	4/9
{A B}	5/9
{A C}	4/9
{B C}	4/9
{A B C}	4/9

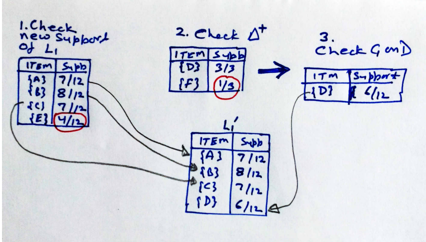
Consider the arrival of $\triangle^+$ more transactions

$T_1 = (A, B, C)$
$T_2 = (A, F)$
$T_3 = (A, B, C, E)$
$T_4 = (A, B, D, F)$
$T_5 = (C, F)$
$T_6 = (A, B, C)$
$T_7 = (A, B, C, E)$
$T_8 = (C, D, E)$
$T_9 = (B, D, E)$

Δ^+

$T_{10} = (B, D)$
$T_{11} = (D, F)$
$T_{12} = (A, B, C, D)$

The first iteration, is as below.



[1] Maintenance of discovered association rules in large databases: An incremental updating technique, *DW Cheung, and J Han, and V Ng, and CY Wong*, International conference on data engineering, pp 106–114, IEEE 1996

Recap: FUP at work (contd...)

The second iteration, is as below.

1. Check new support of L_1

ITEM	Support
{A,B}	6/12
{A,C}	5/12
{B,C}	5/12

2. Check other itemsets in $\triangle^+$

ITEM	Support
{A,D}	1/3
{B,D}	2/3
{C,D}	1/3
{D,F}	1/3

3. Validate C_2 on database D

ITEM	Support
{B,D}	4/12

ITEM	Support
{A,B}	6/12
{A,C}	5/12
{B,C}	5/12

Similarly it is executed for next levels.

FUP_2

- FUP_2 [2] can work for both $\triangle^-$ and $\triangle^+$
- L_k from previous mining result is used for dividing candidate itemset C_k into two parts
  - $P_k = C_k \cap L_k$
  - $Q_k = C_k - P_k$
- Itemset that is frequent in $\triangle^-$, must be infrequent in D^-.
- Further if itemset in Q_k in infrequent in $\triangle^+$ then it is infrequent in D^-.
- This technique helps to effectively reduce number of candidate itemsets.

[2] A general incremental technique for maintaining discovered association rules, *DW Cheung, SD Lee, and B Kao*, Database Systems For Advanced Applications, pp: 185–194, World Scientific-1997

FUP$_2$ at work

D:

Δ^-:
$T_1 = (A, B, C)$
$T_2 = (A, F)$
$T_3 = (A, B, C, E)$

$T_4 = (A, B, D, F)$
$T_5 = (C, F)$
$T_6 = (A, B, C)$

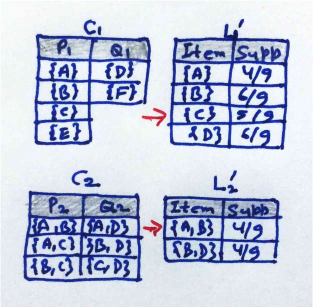
$T_7 = (A, B, C, E)$
$T_8 = (C, D, E)$
$T_9 = (B, D, E)$

Δ^+:
$T_{10} = (B, D)$
$T_{11} = (D, F)$
$T_{12} = (A, B, C, D)$

D': T_4 – T_{12}

Item set	Support
{A}	6/9
{B}	6/9
{C}	6/9
{E}	4/9
{A B}	5/9
{A C}	4/9
{B C}	4/9
{A B C}	4/9

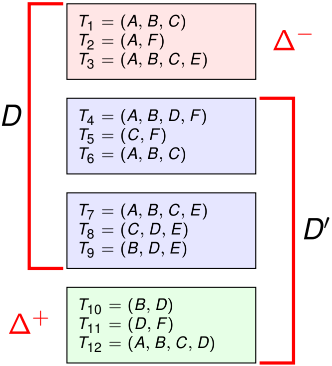
Frequent itemsets of D



- C_1 is set of all items. It is divided in P_i and Q_i
- Being frequent, support for all items in P_i is known. It could be updated using $\triangle^-$ and $\triangle^+$ only.
- Count$(\{A\})_{D'}$ = Count$(\{A\})_D$ − Count$(\{A\})_{\triangle^-}$ + Count$(\{A\})_{\triangle^+}$ $= 6 - 3 + 1 = 4$

FUP$_2$ at work

- In some cases only the scan of $\triangle^-$ and $\triangle^+$ is required.
- For example, Count$(\{F\})_{\triangle^+} -$ Count$(\{F\})_{\triangle^-} = 0$ showing that support of $\{F\}$ can not be improved.
- Consequently, fewer itemsets have to be further scanned
- An iteration finishes when all the itemsets in P_i and Q_i are verified, and new set of frequent itemsets L'_i is generated



FUP$_2$H

Uses hashing for performance improvement

Variations of FUP

- **Update With Early Pruning (UWEP):** Occurrence of potentially huge set of candidate itemset and multiple scans of the database is the issue
  - If a k-itemset is frequent in $_{\Delta^+}$ but infrequent in D', it is not considered when generating C_{k+1}
  - This can significantly reduce the number of candidate itemsets, with the trade-off that an additional set of unchecked itemsets has to be maintained.
- **Utilizing Negative Borders:** Negative border set consists of all itemsets that are closest to be frequent
  - Negative border consists of all itemsets that were candidates of level-vise method but did not have enough support

$$Bd^{-}(L) = C_k - L_k$$

  - Find negative border set for $L = \{\{A\}, \{B\}, \{C\}, \{E\}, \{AB\}, \{AC\}, \{BC\}, \{ABC\}\}$
  - Full scan of dataset is only required when *itemsets outside negative border set* get added to frequent itemsets or negative border set.

Law of large number

$$Prob\left(\left|\frac{x_1 + x_2 + x_3 + ... + x_n}{n} - E(x)\right| \geq \epsilon\right) \leq \frac{var(x)}{n\epsilon^2}$$

Law of large number

$$Prob\left(\left|\frac{x_1 + x_2 + x_3 + ... + x_n}{n} - E(x)\right| \geq \epsilon\right) \leq \frac{var(x)}{n\epsilon^2}$$

- **Markov's inequality**
  When x be a non-negative *random variable*. Then for $a > 0$

$$Prob(x \geq a) \leq \frac{E(x)}{a}$$

Law of large number

$$Prob(|\frac{x_1 + x_2 + x_3 + ... + x_n}{n} - E(x)| \geq \epsilon) \leq \frac{var(x)}{n\epsilon^2}$$

- **Markov's inequality**
  When x be a non-negative *random variable*. Then for $a > 0$

$$Prob(x \geq a) \leq \frac{E(x)}{a}$$

- **Chebyshev's Inequality**
  Let x be a *random variable*. Then for $c > 0$

$$Prob(|x - E(x)| \geq c) \leq \frac{Var(x)}{c^2}$$

Variations of FUP

- **Difference Estimation for Large Itemsets (DELI)[3]:** Uses sampling technique
  - Estimate the difference between old and new frequent itemsets
  - Iff the difference is large, update operation using FUP_2 is performed
  - Let S be m transactions drawn from D^- with replacement, then support of itemset X in D^- is

$$\hat{\sigma_X} = \frac{T_x}{m}.|D^-|$$

where T_x is occurrence count of X in S. For large m we have 100(1-α)% confidence interval [a_x,b_x] with

$$a_x = \hat{\sigma_X} - z_{a/2}\sqrt{\frac{\hat{\sigma_X}(|D^-| - \hat{\sigma_X})}{m}}$$

$$b_x = \hat{\sigma_X} + z_{a/2}\sqrt{\frac{\hat{\sigma_X}(|D^-| - \hat{\sigma_X})}{m}}$$

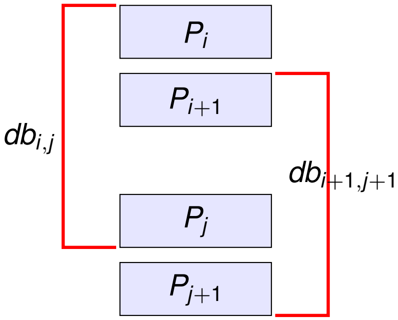
where $z_{a/2}$ is a value such that the area beyond it in standard normal curve is exactly $\alpha/2$

[3] Is sampling useful in data mining? a case in the maintenance of discovered association rules, *SD Lee, D Sau, DW Cheung, W David, and B Kao*, Data Mining and Knowledge Discovery, 2(3), pp 233–262, Springer-1998

Sliding Window Filtering

Partition-Based Algorithm for Incremental Mining:

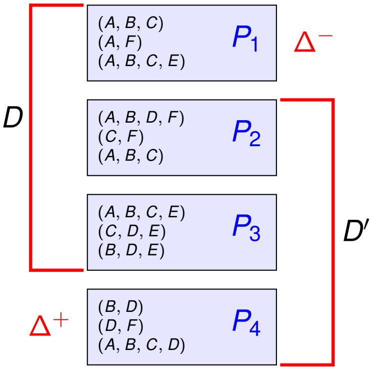
If X is a frequent itemset in a database divided into partitions p_1, p_2, ..., p_n then X must be a frequent itemset in at least one of the partitions



- Uses threshold to generate candidate itemset
- Frequent itemset remains frequent from some P_k to P_n
- A list of 2-itemsets CF is maintained to track possible frequent 2-itemsets.
- Locally frequent 2-itemsets of each partition is added (with its starting partition and supports)
- Scan reduction technique can make one database scan enough

SWF at work

With $S_{min} = 40\%$ generate frequent 2-itemsets



P_1

Item	Start	Count
{A,B}	1	2
{A,C}	1	2
{B,C}	1	2

P_2

Item	Start	Count
{A,B}	1	4
{A,C}	1	3
{B,C}	1	3

P_3

Item	Start	Count
{A,B}	1	5
{A,C}	1	4
{B,C}	1	4
{B,E}	3	2
{C,E}	3	2
{D.E}	3	2

No new 2-itemset added when processing P_2 since no extra frequent 2-itemsets. Moreover, the counts for itemsets {A,B}, {A,C} and {B,C} are all increased.

SWF at work

- Scan reduction technique is used to generate C_k $(k = 2, 3, ..., n)$ using C_2
- C_2 is used to generate the candidate 3-itemsets and its sequential C'_{k1} be utilized to generate C'_k
- C'_3 generated from $C_2 * C_2$ instead of $L_2 * L_2$ will have size greater but near to $|C_3|$
- Second scan would suffice for pruning

Merit of SWF lies in its incremental procedure. There are three sub-steps

- Generating C_2 in $D^- = db^{1,3} - \triangle^-$
- Generating C_2 in $db^{2,4} = D^- + \triangle^+$
- Scanning $db^{2,4}$ once

$db^{1,3} - \Delta^- = D^-$

Itemset	Start	Count
{A,B}	2	3
{A,C}	2	2
{B,C}	2	2
{B,E}	3	2
{C,E}	3	2
{D,E}	3	2

$D^- + \Delta^+ = D'$

Itemset	Start	Count
{A,B}	2	4
{B,E}	3	2
{C,E}	3	2
{D,E}	3	2
{D,B}	4	3

Thank You!

Thank you very much for your attention!

Queries ?