Minimally Infrequent Itemset Mining using Pattern-Growth Paradigm and Residual Trees

Ashish Gupta, Akshay Mittal, Arnab Bhattacharya
{ashgupta,amittal,arnabb}@cse.iitk.ac.in

Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

COMAD'11

19th December, 2011

PROBLEM STATEMENT

- Transaction database $D = \{T\}$ and a set of items I
- Transaction T: A tuple where (tid, X) where tid is the transaction identifier and X is an itemset
- $supp(X, TD) = X.count$, $X.count$ is the number of transactions in TD that contain X
- k-itemset: Set of k items $I_k = \{x_1, x_2, \ldots, x_k\}$
- User defined threshold σ
  - An itemset is frequent if and only if its support is greater than or equal to σ; it is infrequent otherwise

DEFINITION (MINIMALLY INFREQUENT ITEMSET)

An itemset X is said to be minimally infrequent for a support threshold σ if it is *infrequent* and *all* its proper subsets are *frequent*, i.e., $supp(X) < \sigma$ and $\forall Y \subset X, supp(Y) \geq \sigma$.

MOTIVATION

Infrequent Itemsets:

- Mining of negative association rules, *Xindong et al.* [5]
- Statistical disclosure risk assessment, *Haglin et al.* [2]
- Fraud detection, *Haglin et al.* [2]
- Bio-informatics, *Haglin et al.* [2]

Motivation

Infrequent Itemsets:

- Mining of negative association rules, *Xindong et al.* [5]
- Statistical disclosure risk assessment, *Haglin et al.* [2]
- Fraud detection, *Haglin et al.* [2]
- Bio-informatics, *Haglin et al.* [2]

Paradigms

- Candidate generation-and-test paradigm, *Agrawal et al.* [1]
- Pattern-growth paradigm, *Han et al.* [3]

Pattern-growth based algorithms are computationally faster on dense datasets

Motivation

Infrequent Itemsets:

- Mining of negative association rules, *Xindong et al.* [5]
- Statistical disclosure risk assessment, *Haglin et al.* [2]
- Fraud detection, *Haglin et al.* [2]
- Bio-informatics, *Haglin et al.* [2]

Paradigms

- Candidate generation-and-test paradigm, *Agrawal et al.* [1]
- Pattern-growth paradigm, *Han et al.* [3]

Pattern-growth based algorithms are computationally faster on dense datasets

- Main aim: Leverage the pattern-growth paradigm to propose an algorithm IFP_min for mining minimally infrequent itemsets

Example

Tid	Transactions
T_1	F, E
T_2	A, B, C
T_3	A, B
T_4	A, D
T_5	A, C, D
T_6	B, C, D
T_7	E, B
T_8	E, C
T_9	E, D

TABLE: Example database for infrequent itemset mining.

- $\sigma = 2$
- $\{B, D\}$ is an MII since all its subsets, i.e., $\{B\}$ and $\{D\}$, are frequent but it itself is infrequent as its support is 1
- MIIs = $\{\{E, B\}, \{E, C\}, \{E, D\}, \{B, D\}, \{A, B, C\}, \{A, C, D\}, \{A, E\}, \{F\}\}$
- $\{B, F\}$ is not a MII since one of its subsets $\{F\}$ is infrequent as well

Related Work

- Apriori - Candidate generation-and-test paradigm [1]
  - Generated candidate frequent itemsets whose subsets are all frequent
- FP-Growth - Pattern-growth paradigm [3]
  - Depth-first search algorithm
  - Uses the data structure *FP-Tree* used for storing the frequency information of itemsets in a compressed form
  - Faster than Apriori on dense datasets
- MINIT
  - Proposed by *Haglin et al.* [2]
  - Based on SUDA2 algorithm [4] developed for finding unique item-sets (itemsets with no unique proper subsets)
  - Showed that the minimal infrequent itemset problem is NP-complete

IFP-TREE VS FP-TREE

DEFINITION (FLIST)

List of items present in the transaction database sorted in *decreasing* order of their support counts. Used in construction of FP-Tree.

DEFINITION (I-FLIST)

List of items present in the transaction database sorted in increasing order of their support counts. Used in construction of IFP-Tree.

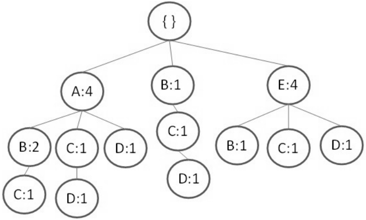


FIGURE: IFP-tree corresponding to the transaction database in Table 1 (T_2-T_9).

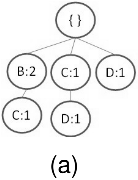
Projected and Residual Trees of an item x

Definition (Projected Tree T_{P_x})

Constructed from the projected database of item x which is the set of transactions containing x.

Definition (Residual Tree T_{R_x})

Constructed from the residual database of item x which is the original database without item x.



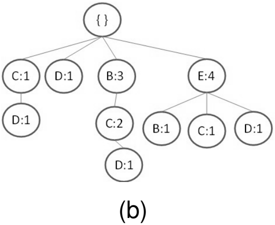


Figure: (a) Projected tree T_{P_A} and (b) Residual tree T_{R_A} of item A for the IFP-tree shown in Figure 1.

MIIs from the Residual Tree

Observation

An itemset S (not containing the item x) is frequent (infrequent) in the residual tree T_{R_x} if and only if the itemset S is frequent (infrequent) in T, i.e.,

$$S \text{ is frequent in } T \Leftrightarrow S \text{ is frequent in } T_{R_x} \ (x \notin S)$$
$$S \text{ is infrequent in } T \Leftrightarrow S \text{ is infrequent in } T_{R_x} \ (x \notin S)$$

Theorem

An itemset S (not containing the item x) is minimally infrequent in T if and only if the itemset S is minimally infrequent in the residual tree T_{R_x} of x, i.e.,

$$S \text{ is minimally infrequent in } T$$
$$\Leftrightarrow S \text{ is minimally infrequent in } T_{R_x} \ (x \notin S)$$

MIIs from the Projected Tree

Observation

An itemset S is frequent (infrequent) in the projected tree T_{P_x} if and only if the itemset obtained by including x in S (i.e., $x \cup S$), is frequent (infrequent) in T, i.e.,

$$x \cup S \text{ is frequent in } T \Leftrightarrow S \text{ is frequent in } T_{P_x}$$
$$x \cup S \text{ is infrequent in } T \Leftrightarrow S \text{ is infrequent in } T_{P_x}$$

Theorem

An itemset $\{x\} \cup S$ is minimally infrequent in T if and only if the itemset S is minimally infrequent in the projected tree T_{P_x} but not minimally infrequent in the residual tree T_{R_x}, i.e.,

$$\{x\} \cup S \text{ is minimally infrequent in } T$$
$$\Leftrightarrow S \text{ is minimally infrequent in } T_{P_x} \text{ and}$$
$$S \text{ is not minimally infrequent in } T_{R_x}$$

EXAMPLE

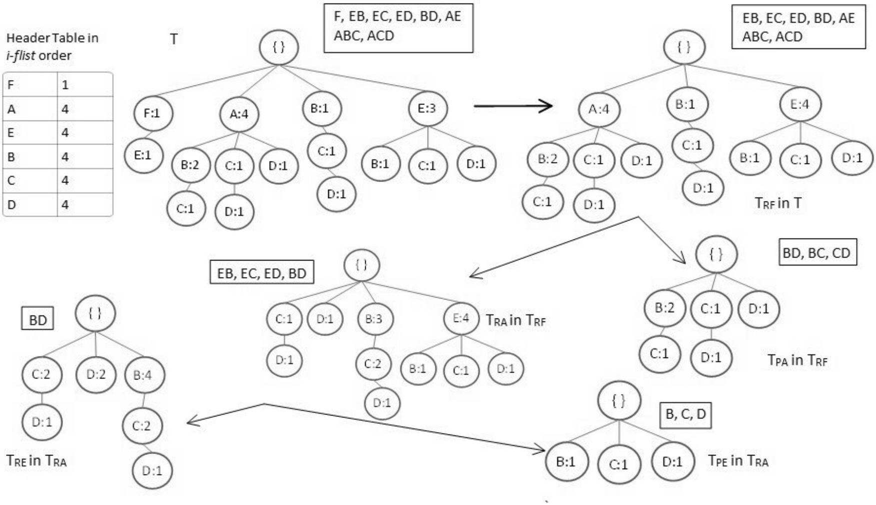


FIGURE: Example for the transaction database in Table 1.

The IFP_Min Algorithm

Step 1

- Infrequent 1-itemsets are trivial MIIs. Pruned and returned.

The IFP_Min Algorithm

Step 1

- Infrequent 1-itemsets are trivial MIIs. Pruned and returned.

Step 2

- Select the least frequent item. Divide the database into two non-disjoint sets – *projected database* and *residual database* – of that item.
- Apply IFP_Min on residual database recursively and obtain its MII
- Apply IFP_Min on projected database recursively and obtain its MII

The IFP_Min Algorithm

Step 1

- Infrequent 1-itemsets are trivial MIIs. Pruned and returned.

Step 2

- Select the least frequent item. Divide the database into two non-disjoint sets – *projected database* and *residual database* – of that item.
- Apply IFP_Min on residual database recursively and obtain its MII
- Apply IFP_Min on projected database recursively and obtain its MII

Step 3

- Return the MIIs of the residual databases as MII of the original tree.
- Check for each itemset in the MII of the projected database whether it is present in the residual databases. If not, append it with the least frequent item and return as MII of the original tree.

THE IFP_MIN ALGORITHM

STEP 1

- Infrequent 1-itemsets are trivial MIIs. Pruned and returned.

STEP 2

- Select the least frequent item. Divide the database into two non-disjoint sets – *projected database* and *residual database* – of that item.
- Apply IFP_Min on residual database recursively and obtain its MII
- Apply IFP_Min on projected database recursively and obtain its MII

STEP 3

- Return the MIIs of the residual databases as MII of the original tree.
- Check for each itemset in the MII of the projected database whether it is present in the residual databases. If not, append it with the least frequent item and return as MII of the original tree.

STEP 4

- Frequent items that do not occur in the projected tree of *x* form MIIs (of length 2) when combined with x individually

MIIs using Apriori

Candidate generation-and-test (Apriori-based) algorithms are computationally faster on sparse datasets

- Consider the iteration where candidate itemsets of length $l+1$ are generated from frequent itemsets of length l.
- From the generated candidate set, itemsets whose support satisfies the minimum support threshold are reported as frequent and the rest are rejected. This rejected set of itemsets constitute the MIIs of length $l+1$.
- For such an itemset, all the subsets are frequent (due to the candidate generation procedure) while the itemset itself is infrequent.
- For experimentation purposes, we label this algorithm as the *Apriori_min* algorithm.

Challenges

- Single threshold is used for generating frequent itemsets irrespective of the length of the itemset
- Support threshold is too high $\Rightarrow$ Less number of frequent itemsets will be generated resulting in loss of valuable association rules
- Support threshold is too low $\Rightarrow$ Large number of frequent itemsets and consequently large number of association rules are generated, thereby making it difficult for the user to choose the important ones

Challenges

- Single threshold is used for generating frequent itemsets irrespective of the length of the itemset
- Support threshold is too high $\Rightarrow$ Less number of frequent itemsets will be generated resulting in loss of valuable association rules
- Support threshold is too low $\Rightarrow$ Large number of frequent itemsets and consequently large number of association rules are generated, thereby making it difficult for the user to choose the important ones

MLMS Model

- Separate thresholds $\sigma_1, \sigma_2, \ldots, \sigma_n$, etc. are assigned to itemsets of different sizes in order to constrain the number of frequent itemsets mined
- Optimizes the number of association rules generated
- σ_k is the minimum support threshold for a *k-itemset* to be frequent

Example of an MLMS Model

Tid	Transactions	Items in *i-flist* order
T_1	A, C, T, W	A, T, W, C
T_2	C, D, W	D, W, C
T_3	A, C, T, W	A, T, W, C
T_4	A, D, C, W	A, D, W, C
T_5	A, T, C, W, D	A, D, T, W,
T_6	C, D, T, B	B, D, T, C

TABLE: Example database for MLMS model.

σ_k	Frequent *k-itemsets*
$\sigma_1 = 4$	{C}, {W}, {T}, {D}, {A}
$\sigma_2 = 4$	{C, D}, {C, W}, {C, A}, {W, A}, {C, T}
$\sigma_3 = 3$	{C, W, T}, {C, W, D}, {C, W, A}, {C, T, A}, {W, T, A}
$\sigma_4 = 2$	{C, W, T, A}, {C, D, W, A}
$\sigma_5 = 1$	{C, W, T, D, A}

TABLE: Frequent *k-itemsets* for database in Table 2.

Terminology

Definition (Prefix-set of a tree)

The prefix-set of a tree is the set of items that need to be included with the itemsets in the tree, i.e., all the items on which the projections have been done. For a tree T, it is denoted by Δ_T.

Definition (Prefix-length of a tree)

The prefix-length of a tree is the length of its prefix-set. For a tree T, it is denoted by $\rho_T = |\Delta_T|$.

Definition (Frequent* itemset)

A k-itemset S is frequent* in T having $\rho_T = p$ if $supp(S, T) \geq \sigma_{k,p} = \sigma_{k+p}$.

Definition (Infrequent* itemset)

A k-itemset S is infrequent* in T having $\rho_T = p$ if $supp(S, T) < \sigma_{k,p} = \sigma_{k+p}$.

USING PROJECTED AND RESIDUAL TREES

THEOREM

An itemset $\{x\} \cup S$ *is frequent* in* T *if and only if* S *is frequent* in the projected tree* T_{P_x} *of* x*, i.e.,*

$$\{x\} \cup S \text{ is frequent* in } T \Leftrightarrow S \text{ is frequent* in } T_{P_x}$$

THEOREM

An itemset S *(not containing* x*) is frequent* in* T *if and only if* S *is frequent* in the residual tree* T_{R_x} *of* x*, i.e.,*

$$S \text{ is frequent* in } T \Leftrightarrow S \text{ is frequent* in } T_{R_x}$$

- Algorithm IFP_MLMS uses these two theorems
- The algorithm is similar to FP-Growth algorithm with the difference of checking for *frequent** itemsets instead of *frequent* itemsets

Experimental Setup

	Number of transactions	Number of items
Real Datasets		
Accidents	340183	468
Connect	67557	129
Mushroom	8124	119
Chess	3196	75
Synthetic Datasets		
T10I4D100K	100000	870
T40I10D100K	100000	942

Table: Details of datasets

IFP_MIN, MINIT AND APRIORI_MIN

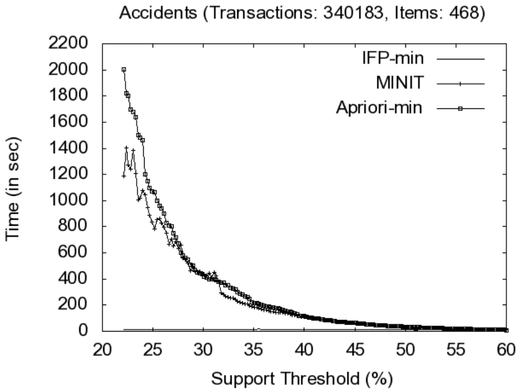


FIGURE: Accident Dataset

- IFP_min outperforms MINIT and Apriori_min for all thresholds

IFP_MIN AND MINIT

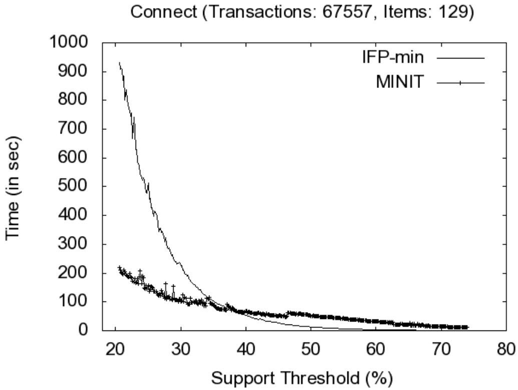


FIGURE: Connect Dataset

- IFP_min outperforms MINIT at higher thresholds while at lower thresholds, MINIT performs better. The neutral threshold point occurs at around 40%

IFP_MIN AND MINIT

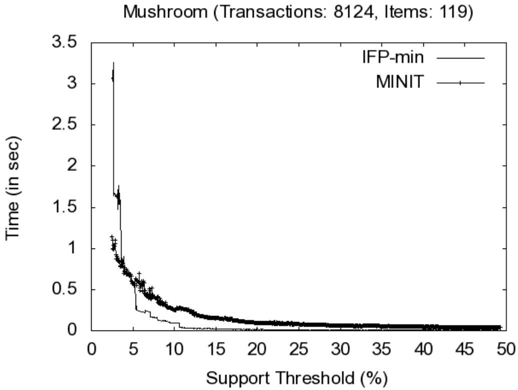


FIGURE: Mushroom Dataset

- IFP_min outperforms MINIT at higher thresholds while at lower thresholds, MINIT performs better. The neutral threshold point occurs at around 5%

IFP_MIN AND MINIT

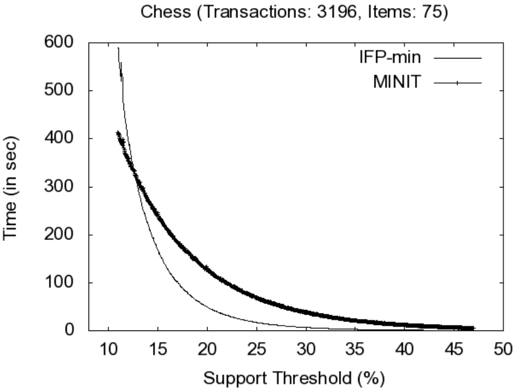


FIGURE: Chess Dataset

- IFP_min outperforms MINIT at higher thresholds while at lower thresholds, MINIT performs better. The neutral threshold point occurs at around 15%

IFP_MIN, MINIT AND APRIORI_MIN

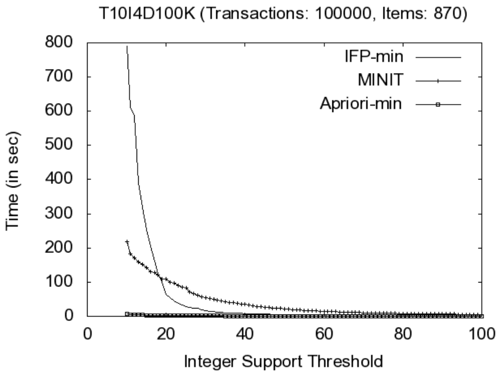


FIGURE: T10I4D100K Dataset

- Apriori_min outperforms both IFP_min and MINIT
- Candidate generation-and-test based algorithms are computationally faster on sparse datasets

IFP_MIN, MINIT AND APRIORI_MIN

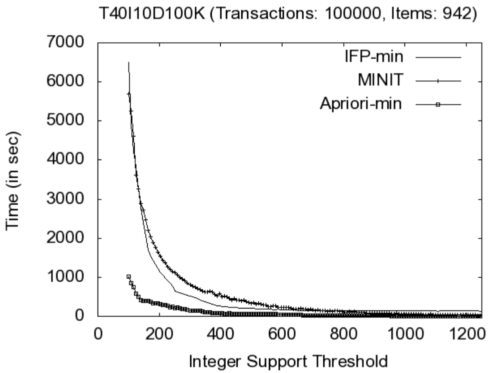


FIGURE: T40I10D100K Dataset

- Apriori_min outperforms both IFP_min and MINIT
- Candidate generation-and-test based algorithms are computationally faster on sparse datasets

ANALYSIS

- For sparse datasets, Apriori performs the best
- For dense and large datasets, due to the presence of many candidates and as MINIT checks for their supports in the database, it lags behind IFP_min for all thresholds
- Dense and small datasets are characterized by a neutral support threshold, $\sigma_{neutral}$
- Below $\sigma_{neutral}$, MINIT algorithm performs better than IFP_min
  - Prunes an item based on the support threshold and length of the itemset in which the item is present (*minimum support property*)
  - As the support thresholds are reduced, the pruning condition becomes activated and leads to reduction in search space.
  - Above the neutral point, the pruning condition is not effective
- Above $\sigma_{neutral}$, IFP_min performs better than MINIT
  - Any candidate MII itemset is checked for set membership in a residual database whereas in MINIT the candidates are validated by computing the support from the whole database.
  - Reduced validation space leads to IFP_min outperforming MINIT

IFP_MLMS and Apriori_MLMS

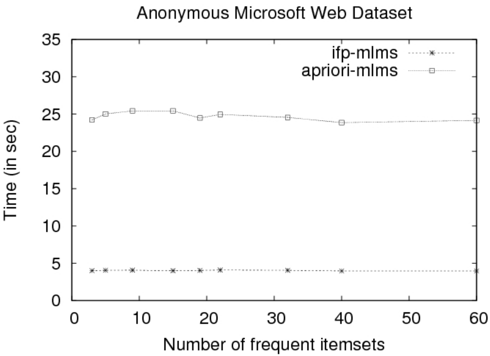


FIGURE: The IFP_MLMS and Apriori_MLMS algorithms on the Anonymous Microsoft Web Dataset.

- Minimum support thresholds for itemsets of different lengths were varied over a distribution window from 2% to 20% at regular intervals

IFP_MLMS and Apriori_MLMS

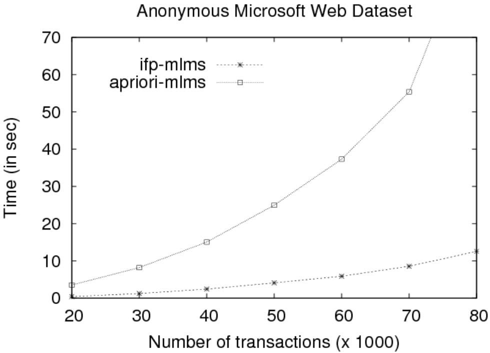


FIGURE: The IFP_MLMS and Apriori_MLMS algorithms on the Anonymous Microsoft Web Dataset.

- Minimum support thresholds kept between 3% to 10%
- IFP_MLMS outperforms existing Apriori_MLMS

Analysis

- Running time of both IFP_MLMS and Apriori_MLMS algorithms are independent of support thresholds for the MLMS model.
- Absence of downward closure property of frequent itemsets in the MLMS model, unlike that of the single threshold model.

Analysis

- Running time of both IFP_MLMS and Apriori_MLMS algorithms are independent of support thresholds for the MLMS model.
- Absence of downward closure property of frequent itemsets in the MLMS model, unlike that of the single threshold model.

Consider an alternative FP-Growth algorithm for the MLMS model that mines all frequent itemsets corresponding to the lowest support threshold σ_{low} and then filters the σ_k frequent k-itemsets to report the frequent itemsets.

- A very large set of frequent itemsets is generated
- Renders the filtering process computationally expensive

Analysis

- Running time of both IFP_MLMS and Apriori_MLMS algorithms are independent of support thresholds for the MLMS model.
- Absence of downward closure property of frequent itemsets in the MLMS model, unlike that of the single threshold model.

Consider an alternative FP-Growth algorithm for the MLMS model that mines all frequent itemsets corresponding to the lowest support threshold σ_{low} and then filters the σ_k frequent k-itemsets to report the frequent itemsets.

- A very large set of frequent itemsets is generated
- Renders the filtering process computationally expensive

Advantages of IFP_MLMS

- Pruning based on σ_{low} in IFP_MLMS ensures that the search space is same for both the algorithms.
- Filtering done in the FP-Growth algorithm is implicitly performed in IFP_MLMS, thus making IFP_MLMS more efficient.

Conclusions

- Introduced a novel algorithm, IFP_min, for mining minimally infrequent itemsets (MIIs).

CONCLUSIONS

- Introduced a novel algorithm, IFP_min, for mining minimally infrequent itemsets (MIIs).
- Proposed an improvement of the Apriori algorithm to find the MIIs.

CONCLUSIONS

- Introduced a novel algorithm, IFP_min, for mining minimally infrequent itemsets (MIIs).
- Proposed an improvement of the Apriori algorithm to find the MIIs.
- Experimental results show that
  - For large dense datasets, it is preferable to use IFP_min algorithm,
  - For small dense datasets, MINIT should be used at low support thresholds and IFP_min should be used at larger thresholds and
  - For sparse datasets, Apriori_min should be used for reporting the MIIs.

Conclusions

- Introduced a novel algorithm, IFP_min, for mining minimally infrequent itemsets (MIIs).
- Proposed an improvement of the Apriori algorithm to find the MIIs.
- Experimental results show that
  - For large dense datasets, it is preferable to use IFP_min algorithm,
  - For small dense datasets, MINIT should be used at low support thresholds and IFP_min should be used at larger thresholds and
  - For sparse datasets, Apriori_min should be used for reporting the MIIs.
- Designed an extension of the algorithm for finding frequent itemsets in the multiple level minimum support (MLMS) model.

CONCLUSIONS

- Introduced a novel algorithm, IFP_min, for mining minimally infrequent itemsets (MIIs).
- Proposed an improvement of the Apriori algorithm to find the MIIs.
- Experimental results show that
  - For large dense datasets, it is preferable to use IFP_min algorithm,
  - For small dense datasets, MINIT should be used at low support thresholds and IFP_min should be used at larger thresholds and
  - For sparse datasets, Apriori_min should be used for reporting the MIIs.
- Designed an extension of the algorithm for finding frequent itemsets in the multiple level minimum support (MLMS) model.
- Experimental results show that IFP_MLMS outperforms the existing candidate-generation-and-test based Apriori_MLMS algorithm.

Conclusions

- Introduced a novel algorithm, IFP_min, for mining minimally infrequent itemsets (MIIs).
- Proposed an improvement of the Apriori algorithm to find the MIIs.
- Experimental results show that
  - For large dense datasets, it is preferable to use IFP_min algorithm,
  - For small dense datasets, MINIT should be used at low support thresholds and IFP_min should be used at larger thresholds and
  - For sparse datasets, Apriori_min should be used for reporting the MIIs.
- Designed an extension of the algorithm for finding frequent itemsets in the multiple level minimum support (MLMS) model.
- Experimental results show that IFP_MLMS outperforms the existing candidate-generation-and-test based Apriori_MLMS algorithm.

THANK YOU!

CONCLUSIONS

- Introduced a novel algorithm, IFP_min, for mining minimally infrequent itemsets (MIIs).
- Proposed an improvement of the Apriori algorithm to find the MIIs.
- Experimental results show that
  - For large dense datasets, it is preferable to use IFP_min algorithm,
  - For small dense datasets, MINIT should be used at low support thresholds and IFP_min should be used at larger thresholds and
  - For sparse datasets, Apriori_min should be used for reporting the MIIs.
- Designed an extension of the algorithm for finding frequent itemsets in the multiple level minimum support (MLMS) model.
- Experimental results show that IFP_MLMS outperforms the existing candidate-generation-and-test based Apriori_MLMS algorithm.

THANK YOU!

Questions?

References

R. Agrawal and R. Srikant.
Fast algorithms for mining association rules in large databases.
In *VLDB*, pages 487–499, 1994.

D. J. Haglin and A. M. Manning.
On minimal infrequent itemset mining.
In *DMIN*, pages 141–147, 2007.

J. Han, J. Pei, and Y. Yin.
Mining frequent patterns without candidate generation.
In *SIGMOD*, pages 1–12, 2000.

A. M. Manning, D. J. Haglin, and J. A. Keane.
A recursive search algorithm for statistical disclosure assessment.
Data Mining Know. Discov., 16:165–196, 2008.

X. Wu, C. Zhang, and S. Zhang.
Efficient mining of both positive and negative association rules.
ACM Trans. Inf. Syst., 22(3):381–405, 2004.