Finding Recent Frequent Itemsets Adaptively over Online Data Stream

Yueting Chen

Outline

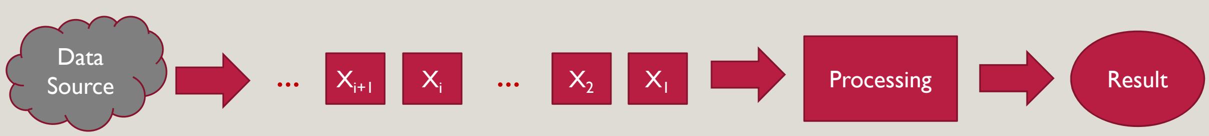
- Introduction
  - Data Stream
  - Related Works
- Preliminaries
- Finding recent frequent itemsets
  - Count estimations of an itemset
  - *estDec* Method
- Experiments
- Conclusions

Introduction

Data Stream & Related Work

Data Stream

- A massive unbounded sequence of data elements
  - Continuously generated
  - At a rapid rate
  - More likely to be changed as time goes by



Challenges

- Each data event should be examined ***at most once***.
- Memory usage for data stream analysis should be ***restricted finitely***.
- Newly generated data elements should be processed ***as fast as possible*****.**
- Up-to-date analysis result of a data stream should be ***instantly available*** when requested

Data Stream Types

- Offline Data Stream
  - Application: data warehouse system
  - Batch processing model
    - Process a number of new transactions together.
  - Up-to-date result only available after a batch process is finished.
  - The granularity of generating results depends on the batch size.
- Online Data Stream
  - Application: network monitoring
  - Batch processing model is ***not*** applicable.
  - Tradeoffs between processing time & mining accuracy without any fixed granule.

Related Works

- *Lossy Counting* algorithm
- *SWF* algorithm

Lossy Counting algorithm

- Two parameters:
  - Minimum support
  - Maximum allowable error ε
- Batch Process model with a fixed buffer
- Use a data structure(*D*) to maintain the previous result
  - Containing a set of entries of form (e, *f*, *Δ*) → *Maximum possible error count*

    e → *itemset*, *f* → *count*
- Update method (for each itemset in a batch):
  - If itemset e not in *D*, insert a new entry.
  - Else *f* ←*f* + *(new count)*
    - *If f+Δ* < εxN, then prune this entry from D.
  - Δ ←⌊εxN'⌋, N' number of transactions that were processed up to the latest batch.

Lossy Counting algorithm

- Can not identify the recent change of stream

SWT Algorithm

- Use sliding window to find frequent itemsets
  - Each window composed of a sequence of partitions.
  - Each partition maintains a number of transactions.
  - Maintain candidate 2-itemsets separately
- When the window is advanced
  - Disregard oldest partition
  - Adjust the candidate 2-itemsets
  - Generate all possible candidate itemsets
  - Generate new frequent itemsets by scanning all the transactions in the window

SWT Algorithm

- Still use the batch processing model
- Candidate generation takes time.

Objective

- Finding recent frequent itemsets ***adaptively*** over ***online*** data stream
  - Examine each transaction in data stream one-by-one.
  - Without candidate generation
  - Consider information differentiation
  - Minimize the total number of significant itemsets in memory.

Preliminaries

To make life easier

Formal Definitions

- Let $I=\{i_1, i_2, \ldots , i_n\}$ be a ***set of current items***
- An ***itemset e*** is a set of items such that $e \in (2^I - \{\emptyset\})$ where 2^I is the power set of I. The ***length |e| of an itemset*** e is the number of items that form the itemset and it is denoted by an |e|-itemset. An itemset $\{a,b,c\}$ is denoted by *abc*.
- A ***transaction*** is a subset of I and each transaction has a unique transaction identifier *TID*. A transaction generated at the *kth* turn is denoted by T_k.
- When a new transaction T_k is generated, the current ***data stream*** D_k is composed of all transactions that have ever been generated so far i.e., $D_k = \langle T_1, T_2, \ldots , T_k \rangle$ and the ***total number of transactions in*** $\boldsymbol{D_k}$ is denoted by $|D|_k$.

Decay

- Goal: We want to concentrate on most recently generated transactions.
- Decay unit
  - determines the chunk of information to be decayed together.
- Decay rate
  - the reducing rate of a weight for a fixed decay-unit
  - Decay-base ***b*** *(b > 1)*
    - Determines decay the amount of weight reduction per a decay-unit.
  - Decay-base-life ***h***
    - defined by the number of decay-units that makes the current weight be b^{-1}
  - Decay rate ***d***

$$d = b^{-(1/h)} \qquad (b>1,\ h\geq 1,\ b^{-1}\leq d<1)$$

Decay (cont'd)

- Theorem 1. Given a decay rate $d = b^{-(1/h)}$ $(b>1, h\geq 1, b^{-1}\leq d<1)$, the total number of transactions $|D|_k$ in the current data stream D_k is found as follows:

$$|D|_k = \begin{cases} 1 & \text{if } k=1 \\ |D|_{k-1} \times d + 1 & \text{if } k \geq 2 \end{cases}$$

- The value of $|D|_k$ converges to $1/(1-d)$ as the value k increases infinitely.

We'll skip proof here.

Finding recent frequent itemsets

Count Estimation & estDec Method

Finding recent frequent itemsets

- Key issue:
  - Avoid candidate generation.
- Two approaches
  - Use estimated count instead of real count.
  - Use tree structure.

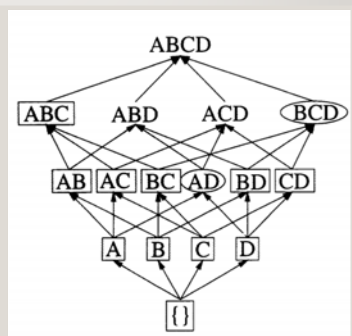


Figure 2: An itemsets lattice.

- Basic idea
  - Use monitoring lattice (a prefix-tree lattice structure)
  - A node in a monitoring lattice contains an item and it denotes an itemset composed of items that are in the nodes of its path from the root.

Count Estimation of an Itemset (Definitions)

- For an *n*-itemset e ($n \geq 2$):
  - A set of its ***subsets*** P(e) is composed of all possible itemsets that can be generated by one or more items of the itemset e $P(e)=\{\alpha|\ \forall\alpha \quad \text{s.t. } \alpha\in 2^e\text{-}\{e\} \text{ and } \alpha\neq\emptyset\}.$
  - A set of its ***m-subsets*** $P_m(e)$ is composed of those itemsets in *P(e)* that have *m* items ($m<n$)

    $$P_m(e)=\{\alpha|\ \forall\alpha \quad \text{s.t.}\alpha\in P(e) \text{ and } |\alpha|=m\}$$
  - A set of ***counts for its m-subsets*** $P_m^c(e)$ is composed of the distinct counts of all itemsets in $P_m(e)$

    $$P_m^C(e)=\{C(\alpha)|\ \forall\alpha \quad \text{s.t. } \alpha\in P_m(e)\}$$ *C(e) denotes the count of an itemset e over a data stream.*
- For two itemsets e_1 and e_2
  - A ***union-itemset*** $e_1 \cup e_2$ is composed of all items that are members of either e_1 or e_2
  - An ***intersection-itemset*** $e_1 \cap e_2$ is composed of all items that are members of both e_1 and e_2.

Count Estimation of an Itemset (Observations)

- Observation:
  - The count of an itemset depends on how often its items appear together in each transaction.
- The possible range of the count of an itemset identified by two extreme distributions
  - LED: *least exclusively distributed*
    - items appear together in as many transactions as possible.
  - MED: *most exclusively distributed*
    - items appear exclusively as many transactions as possible.

Count Estimation of an Itemset (Estimation)

- Estimate the maximum count $C^{max}(e)$
- Fact:
  - If all of e's subsets are LED, then $C^{max}(e)$=smallest value among the counts of its subsets
- Estimation:
  - Use (*n-1*)-subsets to estimate $C^{max}(e)$
  - $C^{max}(e) = \min(P_{n-1}^{C}(e))$ → The set of counts for its (n-1)-subsets

Count Estimation of an Itemset (Estimation)

- For itemset e_1 and e_2 ,the minimum count of their union-itemset:

$$C^{min}(e_1 \cup e_2) = \begin{cases} max(0, C(e_1) + C(e_2) - C(e_1 \cap e_2)) & \text{if } e_1 \cap e_2 \neq \varnothing \\ max(0, C(e_1) + C(e_2) - |D|) & \text{if } e_1 \cap e_2 = \varnothing \end{cases}$$

$|D|$: # of transactions in D

- For each distinct pair (α_i, α_j) of its (*n-1*)-subsets (α_i and $\alpha_j \in Pn\text{-}1(e)$), the count of their union-itemset $\alpha_i \cup \alpha_j$ can be estimated.
- Among the estimated counts for the itemset e, the largest count is the guaranteed appearance count (the minimum count)
- Thus: $C^{min}(e) = max(\{C^{min}(\alpha_i \cup \alpha_j) \mid \forall \alpha_i, \alpha_j \in P_{n-1}(e) \text{ and } i \neq j\})$

Count Estimation of an Itemset (Estimation)

- The maximum count $C^{max}(e)$ of an itemset e is used as the estimated count of the itemset
- The difference between $C^{max}(e)$ and $C^{min}(e)$ be the ***estimation error*** *E*(e) of the itemset

estDec Method (Basic Idea)

- An itemset which has much less support than a predefined minimum support is not necessarily monitored
- The insertion of a new itemset can be delayed until it can possibly be a frequent itemset in the near future.
- When the estimated support of a new itemset is large enough, it is regarded as a ***significant itemset*** and it is inserted to a monitoring lattice
- If current support of a itemset becomes much less than a predefined minimum support, it can be eliminated from the monitoring lattice.

estDec Method (Notations)

- Every node in a monitoring lattice maintains a triple ***(cnt, err, MRtid)*** for a corresponding itemset e.
  - cnt: The ***count*** of the itemset e
  - err: The ***maximum error count*** of the itemset e
  - MRtid: the ***transaction identifier*** of the ***most recent transaction*** that contains the itemset e

estDec Method (Algorithm Outline)

- Process unit: *transaction*
- Four phases:
  - I. Parameter updating phase
  - II. Count updating phase
  - III. Delayed-insertion phase
  - IV. Frequent item selection phase

estDec Method (Phase I. Parameter Updating)

- Update the total number of transactions in the current data stream $|D|_k$
  - $|D|_k = |D|_{k-1} \times d + 1$

estDec Method (Phase II. Count Updating)

- Update the counts of those itemsets in a monitoring lattice that appear in the new transaction.
- Previous triple: (cnt_{pre}, err_{pre}, $MRtid_{pre}$)
- Update triple: (cnt_k, err_k, $MRtid_k$)
  - $cnt_k = cnt_{pre} \times d^{(k-MRtid_{pre})} + 1$
  - $err_k = err_{pre} \times d^{(k-MRtid_{pre})}$
  - $MRtid_k = k$
- Pruning: if $\frac{cnt_k}{|D|_k} < S_{prn}$
  - Exception: 1-itemset will not be pruned, since we need the count for estimations.
  - S_{prn}: threshold for pruning. ($S_{prn} < S_{min}$, S_{min}: minimum support)

estDec Method (Phase III. Delayed-insertion)

- When to insert ?
- A new 1-itemset
  - inserted to a monitoring lattice without any estimation process.
- Estimated support of an *n*-itemset > S_{ins} (*n≥2, not monitored before*)
  - Use estimated value $C^{max}(e)$
  - If any of its (|e|-*1*)-subsets in $P_{n-1}(e)$ is not monitored, $C^{max}(e) = 0$, stop estimation.
  - S_{ins}: threshold for delayed-insertion ($S_{ins} > S_{min}$)
- cnt: $C^{max}(e) = \min(P_{n-1}^{C}(e))$
- Can we estimate *cnt* using other information?

$C^{max}(e) = \min(P_{n-1}^{C}(e))$

estDec Method (Phase III. Delayed-insertion)

- When an itemset e is inserted, all of its (|e|-1)-subsets should be monitored in advance.
  - The actual count is maximized when these |e|-1transactions are most recently generated.
  - The decayed count of the itemset e for the insertion of its subsets by these recent |e|-*1* transactions:
    - $cntt\_for\_subsets = d^{|e|-1} + d^{|e|-2} + \ldots + d + 1 = \{1 - d^{(|e|-1)}\}/(1 - d)$
  - The maximum possible decayed count of the itemset e before the recent |e|-*1* transactions:
    - $max\_cnt\_before\_subsets = S_{ins} * \{|D|_{k-(|e|-1)}\} * d^{(e-1)}$
  - Thus, the upper bound of its actual count:
    - $C^{upper}(e) = max\_cnt\_before\_subsets + cnt\_for\_subsets$
- Update the inserted triple: (cnt_k, err_k, $MRtid_k$)
  - $cnt_k = min\{C^{max}(e), C^{upper}(e)\}$
  - $err_k = E(e) = cnt_k - C^{min}(e)$
  - $MRtid_k = k$

estDec Method (Phase IV. Selection)

- Performed only when the mining result of the current data set is required
- an itemset e is frequent if its current support S is greater than minimum support *Smin*.
  - $S = \{cnt \times d^{(k-MRtid)}\} / |D|_k$
  - Current support error $E = \{err \times d^{(k-MRtid)}\} / |D|_k$

$L_k = \varnothing$;
for all itemset $e \in ML$ {
$cnt = cnt \times d^{(k-MRtid)}$; $err = err \times d^{(k-MRtid)}$; $MRtid = k$;
if $(cnt/|D|_k) \geq S_{min}$
$L_k = L_k \cup \{e\}$;

estDec Method (cont'd)

- ***force-pruning***
  - All insignificant itemsets can be pruned together by examining the current support of every itemset in the monitoring lattice.
  - Can be done periodically

Experiments

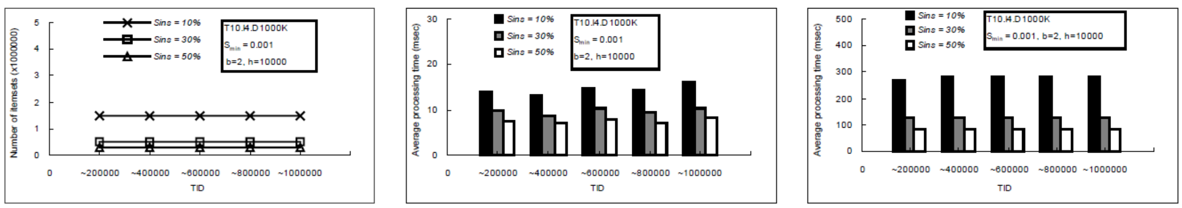
Just show the results

Experiments (Environment)

- Two generated dataset:
  - *T10.I4.D1000K*
  - *T5.I4.D1000K-AB*
- Environment
  - 1.8GHz Pentium PC machine
  - 512MB main memory
  - Linux 7.3
  - All programs are implemented in C

Experiments (Results)

- a) memory usage:
  - The memory usage remains the same. (delayed-insertion and pruning)
- b) & c) average processing time.
  - As the value of S_{ins} is increased, the average processing time is decreased. (smaller search space)



(a) (b) (c)

Figure 2. Performance of the *estDec* method for the data set *T10.I4.D1000K*

Experiments (Results)

- Use *average support error to* model the relative accuracy.

$$ASE(R_2|R_1) = \{ \sum_{e_l \in R_1 - R_1 \cap R_2} S_1(e_l) + \sum_{e_l \in R_1 \cap R_2} (|S_2(e_l) - S_1(e_l)|) + \sum_{e_l \in R_2 - R_1 \cap R_2} S_2(e_l) \} / |R_1|$$

- Measure: $ASE(R_{estDec}|R_{dApriori})$
- dApriori:
  - *Apriori* algorithm with the decay mechanism proposed
- As S_{ins} becomes smaller, more itemsets are maintained in a monitoring lattice, which makes the mining result of the *estDec* method be more accurate.

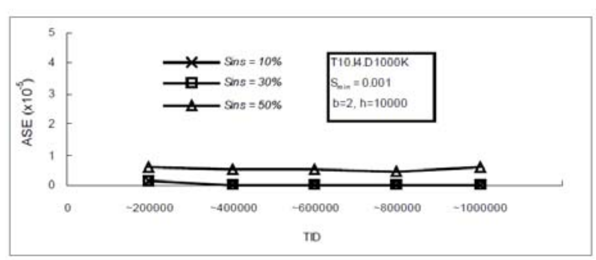


Figure 3. Accuracy of mining results

Experiments (Results)

- *T5.I4.D1000K-AB* (composed of two consecutive subparts, no common items between)
  - Part A: a set of 500,000 transactions generated by an item set *A*
  - Part B: a set of 500,000 transactions generated by an item set *B*
- ***coverage rate CR(X)*** $CR(X) = \frac{\#\text{ of frequent itemsets induced by an item set } X}{|R|} \times 100(\%)$

- As decay-base-life *h* becomes smaller,
- ***OR*** $d = b^{-(1/h)}$
- As decay-base *b* becomes larger
- the *estDec* method adapts more rapidly the transition of information between the two subparts of the data set.

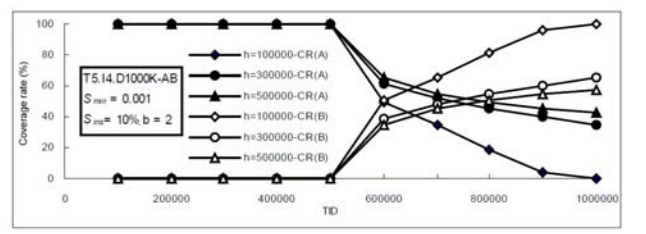


Figure 4. Coverage rate for the data set *T5.I4.D1000K-AB*

Conclusion

Finally …

Conclusion

- Proposed *estDec* method
  - Finds recent frequent itemsets over an online data stream
  - Decay the weight of old transactions as time goes by.
- Advantages
  - The recent change of information in a data stream can be ***adaptively*** reflected to the current mining result
  - The weight of information in a transaction of a data stream is gradually reduced as time goes by
  - The reduction rate can be flexibly controlled.
  - No transaction needs to be maintained physically
- Disadvantages
  - Parameters are hard to determine: Smin, Sprn, Sins, b, h

Thanks

Q&A