Association Analysis

OUTLINE

1. Introduction to market-basket analysis
2. Algorithms for frequent itemset mining
    - Sequential algorithms: A-Priori.
    - Approaches to handle large inputs.

What will we learn?

* Generalization of the Frequent Items problem considering more general patterns (itemsets)
* Challenges to be faced
  - Huge number of potential candidates
    ⇒ we must avoid enumerating all patterns to identify the frequent ones
  - Efficiency for large inputs
  - Control the output size

Introduction to market-basket analysis

Market-basket analysis

DATA:

- A large set of items: e.g., products sold in a supermarket
- A large set of baskets: e.g., each basket represents what a customer bought in one visit to the supermarket

GOAL: Analyze data to extract

- Frequent itemsets: subsets of items that occur together in a (surprisingly) high number of baskets
- Association rules: correlations between subsets of items.

Popular example of association rule: customers who buy diapers and milk are likely to also buy beer

Itemsets and association rules

Dataset $T = \{t_1, t_2, \ldots, t_N\}$ of N transactions (i.e., baskets) over a set I of d items, with $t_i \subseteq I$, for $1 \leq i \leq N$.

Definition (Itemset and its support)

An itemset is a subset $X \subseteq I$ and its support w.r.t. T, denoted by $\mathrm{Supp}_T(X)$, is the *fraction* of transactions of T that contain X (i.e., $|T_X|/N$, where $T_X \subseteq T$ is be the subset of transactions that contain X).

Definition (Association rule and its support and confidence)

An association rule is a rule $r : X \rightarrow Y$, with $X, Y \subset I$, $X, Y \neq \emptyset$, and $X \cap Y = \emptyset$. Its support and confidence w.r.t. T, denoted by $\mathrm{Supp}_T(r)$ and $\mathrm{Conf}_T(r)$, respectively, are defined as

$$\begin{aligned} \mathrm{Supp}_T(r) &= \mathrm{Supp}_T(X \cup Y) \\ \mathrm{Conf}_T(r) &= \mathrm{Supp}_T(X \cup Y)/\mathrm{Supp}_T(X). \end{aligned}$$

X and Y are called, respectively, the rule's antecedent and consequent.

Example

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

$N = 5$ T

$X = \{Milk, Diaper, Beer\}$

$Supp_T(X) = 2/5$

Association rule $r = \{Milk, Diaper\} \rightarrow \{Beer\}$

$Supp_T(r) = 2/5$ $\quad Conf_T(r) = (2/5)/(3/5) = 2/3$

Rigorous formulation of the problems

Given the dataset T of N transactions over I, and given a *support threshold* minsup $\in (0, 1]$, and a *confidence threshold* minconf $\in (0, 1]$, the following two objectives can be pursued:

1. Discover all itemsets $X \neq \emptyset$ with $\text{Supp}_T(X) \geq \text{minsup}$.
   They are called frequent itemsets w.r.t. T and minsup.

2. Discover all association rules r such that $\text{Supp}_T(r) \geq \text{minsup}$ and $\text{Conf}_T(r) \geq \text{minconf}$.

Remark: we will focus on the first problem since:

- The discovery of frequent itemsets is a first step (often computationally heavier) towards the discovery of association rules.
- Many considerations made for frequent itemsets generalize to other domains where frequent patterns are relevant.

Example (minsup = minconf = 0.5)

Dataset T	
TID	Items
1	ABC
2	AC
3	AD
4	BEF

$2^6 - 1 = 63$ itemsets

Frequent Itemsets	
Itemset	Support
A	3/4
B	1/2
C	1/2
AC	1/2

Association Rules		
Rule	Support	Confidence
A $\rightarrow$ C	1/2	2/3
C $\rightarrow$ A	1/2	1

$\frac{1}{2} \cdot \frac{4}{3} = \frac{2}{3}$

Applications

Association analysis is one of the most prominent data mining tasks

1. Analysis of true market baskets. Chain stores keep TBs of data about what customers buy. Frequent itemsets and association rules can help the store lay out products so to "tempt" potential buyers, or decide marketing campaigns.

2. Detection of plagiarism. Consider documents (items) and sentences (transactions). For a given transaction t, its constituent items are those documents where sentence t occurs. A frequent pair of items represent two documents that share a lot of sentences ($\Rightarrow$ possible plagiarism).

3. Analysis of biomarkers. Consider transactions associated with patients, where each transaction contains, as items, biomarkers and diseases. Association rules can help associate a particular disease to specific biomarkers.

Observations

- Support and confidence measure the interestingness of a pattern (itemset or rule), and the thresholds *minsup* and *minconf* define which patterns must be considered interesting.
- Ideally, we would like to discover patterns *unlikely to be seen in a random dataset* (hypothesis testing setting). However, what is a *random dataset*?
- The choice of minsup and minconf directly influences
  - Output size: low thresholds may yield too many patterns (possibly exponential in the input size) which become hard to exploit effectively.
  - False positive/negatives: low thresholds may yield a lot of uninteresting patterns (false positives), while high thresholds may miss some interesting patterns (false negatives).

Potential output explosion

Proposition (proof omitted)

For set I of d items:

- **Number of distinct non-empty itemsets** $= 2^d - 1$
- **Number of distinct association rules** $= 3^d - 2^{d+1} + 1$.

prove it as exercise

As a consequence:

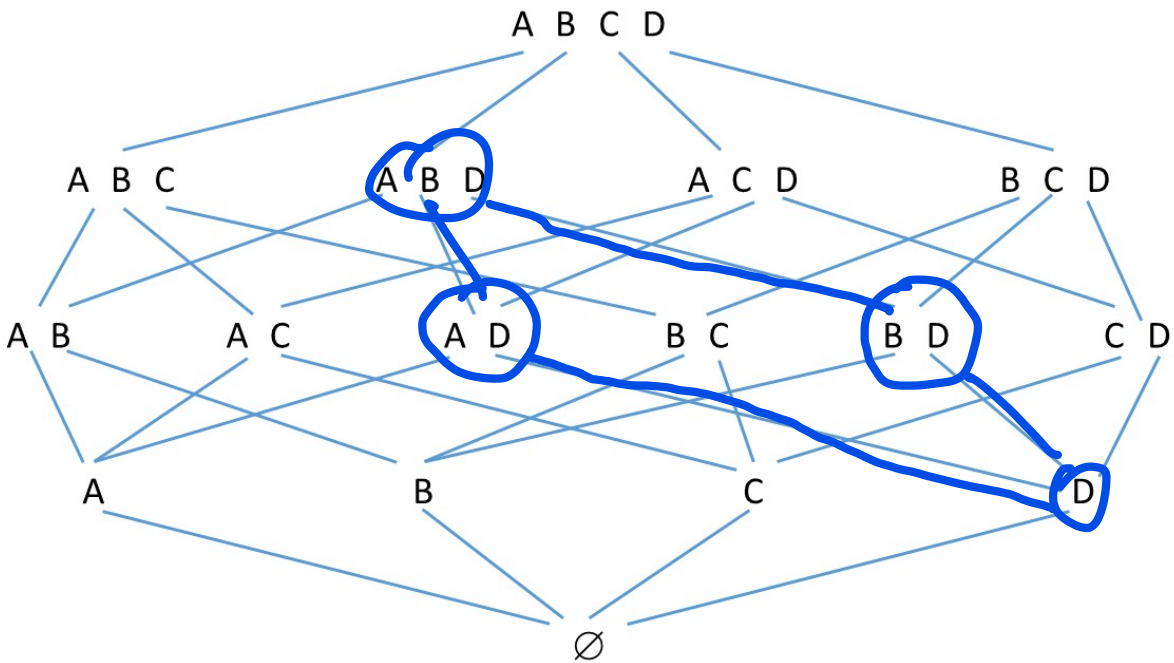
- Enumeration of all itemsets/rules (to find the interesting ones) is out of question even for small sizes (say $d > 40$)
- Complexity analysis w.r.t. input size alone might be not meaningful if output is large!

  $\Rightarrow$ We consider efficient strategies those that require *time/space polynomial in both the input and the output sizes*.

Algorithms for frequent itemset mining

Lattice of Itemsets

- The family of itemsets under $\subseteq$ forms a lattice:
  - Partially ordered set
  - For each two elements X, Y: unique least upper bound $(X \cup Y)$ and a unique greatest lower bound $(X \cap Y)$.
- The lattice can be represented through the Hasse diagram



Anti-monotonicity of support

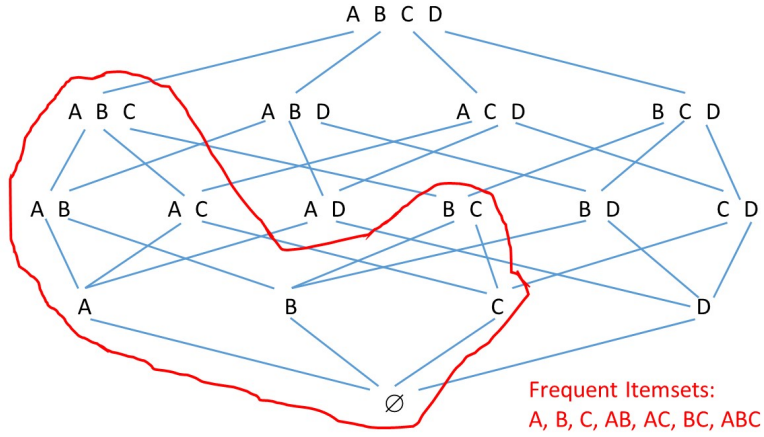
Property (Anti-monotonicity of support): For every $X, Y \subseteq I$

$$X \subseteq Y \Rightarrow \mathrm{Supp}_T(X) \geq \mathrm{Supp}_T(Y).$$

Important consequences: For a given support threshold

1. X is frequent $\Rightarrow$ every $W \subseteq X$ is frequent (**downward closure**)
2. X is not frequent $\Rightarrow$ every $W \supseteq X$ is not frequent

$\Rightarrow$ frequent itemsets form a sublattice closed downwards

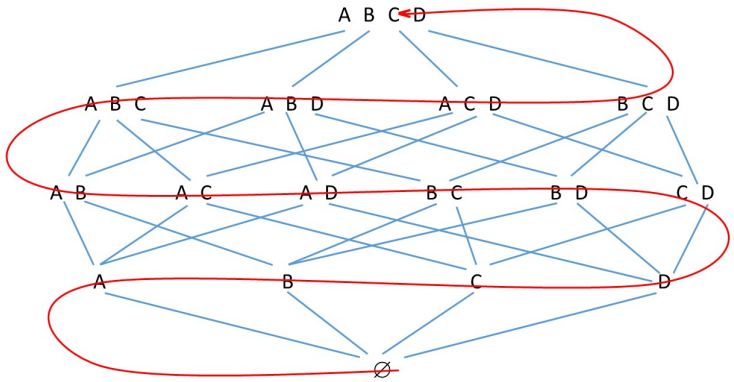


Efficient mining of Frequent Itemsets

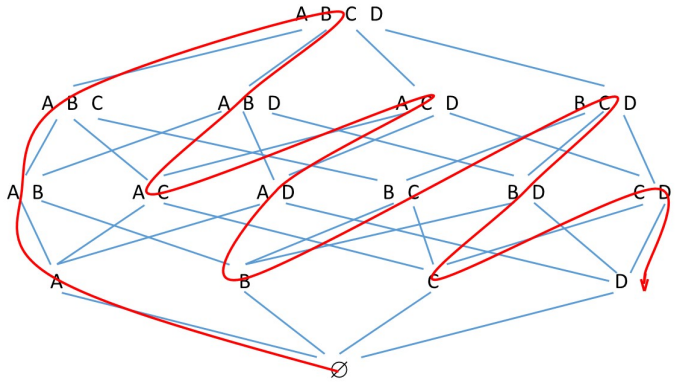
Key objective: Careful exploration of the lattice of itemsets exploiting anti-monotonicity of support

Two main approaches:

Breadth First



Depth First



A-Priori algorithm

A-Priori is a popular, paradigmatic data mining algorithm proposed by Agrawal and Srikant in 1994 at VLDB [AS94]

In 2006 it was included in the Top-10 data mining algorithms.

Uses the breadth-first approach.

A-Priori algorithm: high-level strategy

$T = \{$ N transactions over $I \}$ minsup

$F_k = \{$ Frequent Itemsets of length k w.r.t. T, minsup $\}$

$k \geq 1$

* Computes F_1
* For every $k > 1$
  - Uses F_{k-1} to compute a set $C_k \supseteq F_k$ of CANDIDATES

    aim: small $|C_k|$ using anti-monotonicity of support, and computational efficiency
  - Extract F_k from C_k by computing supports

A-Priori algorithm: pseudocode

Input Dataset T of N transactions over I, minsup

Output $\{(X, \mathrm{Supp}_T(X)) \ : \ X \subseteq I \wedge \mathrm{Supp}_T(X) \geq \mathrm{minsup}\}$

Assumptions and notation

- We assume that transactions/itemsets are represented as sorted vectors, based on some total ordering of the items.
- For every itemset $X \subseteq I$, define its absolute support as the number of transactions that contain X

$$\sigma(X) = \mathrm{Supp}(X) \cdot N$$

A-Priori algorithm: pseudocode

$k \leftarrow 1$

Compute $F_1 = \{i \in I \,;\, \mathtt{Supp}(\{i\}) \geq \mathtt{minsup}\}$

Compute $O_1 = \{(X, \mathtt{Supp}(X)) \,;\, X \in F_1\}$

} use a Hash Table

repeat

  $k \leftarrow k + 1$

  $C_k \leftarrow$ APRIORI-GEN(F_{k-1}) /* *Candidates* */

  for each $c \in C_k$ **do** $\sigma(c) \leftarrow 0$

  for each $t \in T$ **do**

    for each $c \in C_k$ **do**

      if $c \subseteq t$ **then** $\sigma(c) \leftarrow \sigma(c) + 1$

  $F_k \leftarrow \{c \in C_k \,:\, \sigma(c) \geq N \cdot \mathtt{minsup}\}$;

  $O_k \leftarrow \{(X, \sigma(X)/N) \,;\, X \in F_k\}$

until $F_k = \emptyset$

return $\bigcup_{k \geq 1} O_k$

APRIORI-GEN(F): high-level strategy

→ e.g. $F = F_{k-1}$

2 PHASES

1) CANDIDATE GENERATION: merge pairs $X, Y \in F_{k-1}$ sharing all but the last item

E.g. $X = ABCD$ $Y = ABCE \Rightarrow X \cup Y = ABCDE$

2) CANDIDATE PRUNING: remove all candidates that contain, as a subset, at least a non-frequent itemset of length $k-1$

E.g. if $ACDE \notin F_{k-1} \Rightarrow ABCDE$ is removed

Obs. Pruning is done "A PRIORI" without computing supports, but exploiting only anti-monotonicity

APRIORI-GEN(F): pseudocode

APRIORI-GEN Assumes that all itemsets in F have the same length

Let $\ell - 1$ be the size of each itemset in F
$\Phi \leftarrow \emptyset$
/* *Candidate Generation* */
for each $X, Y \in F$ s.t. $X \neq Y \wedge X[1 \ldots \ell-2] = Y[1 \ldots \ell-2]$ **do**
  add $X \cup Y$ to Φ // $|X \cup Y| = \ell$
/* *Candidate Pruning* */
for each $Z \in \Phi$ **do**
  for each $Y \subset Z$ s.t. $|Y| = \ell - 1$ **do**
    if ($Y \notin F$) **then** {remove Z from Φ; exit inner loop}
return Φ

Observation: no itemset is generated twice.

Example

DATASET T	
TID	ITEMS
1	ACD
2	BEF
3	ABCEF
4	ABF

- $N = 4$ **transactions**, $d = 6$ **items**.
- Let's fix $\text{minsup} = 0.5$.
- **Observation:** a brute force strategy would compute the support of $2^d - 1 = 63$ itemsets.

Example

DATASET T	
TID	ITEMS
1	ACD
2	BEF
3	ABCEF
4	ABF

ITEMSET	SUPPORT
A	3/4
B	3/4
C	1/2
~~D~~	~~1/4~~
E	1/2
F	3/4

Set F_1	
ITEMSET	SUPPORT
A	3/4
B	3/4
C	1/2
E	1/2
F	3/4

We computed support for 6 itemsets

Example

DATASET T	
TID	ITEMS
1	ACD
2	BEF
3	ABCEF
4	ABF

Set C_2		
ITEMSET		SUPPORT
After Generation	After Pruning	
AB	AB	1/2
AC	AC	1/2
~~AE~~	~~AE~~	~~1/4~~
AF	AF	1/2
~~BC~~	~~BC~~	~~1/4~~
BE	BE	1/2
BF	BF	3/4
~~CE~~	~~CE~~	~~1/4~~
~~CF~~	~~CF~~	~~1/4~~
EF	EF	1/2

We computed supports for 10 itemsets

Example

Set F_2	
ITEMSET	SUPPORT
AB	1/2
AC	1/2
AF	1/2
BE	1/2
BF	3/4
EF	1/2

DATASET T	
TID	ITEMS
1	ACD
2	BEF
3	ABCEF
4	ABF

Set C_3		
ITEMSET		SUPPORT
After Generation	After Pruning	
ABC		
ABF	ABF	1/2
ACF		
BEF	BEF	1/2

We computed support for 2 itemsets

Example

Set F_3	
ITEMSET	SUPPORT
ABF	1/2
BEF	1/2

$C_4 = \emptyset \Rightarrow F_4 = \emptyset \Rightarrow$

the algorithm stops

Observations

* Altogether we computed supports for 6+10+2 = 18 itemsets out of the 63 non empty itemsets
* Although ABF and BEF share 2 items there is no point generating their union as a candidate itemset (ABF ∪ BEF = ABEF) because if it was frequent then ABE and ABF would be frequent hence ABEF would be generated using the standard procedure

Correctness of A-Priori

Theorem (Correctness)

The A-Priori algorithm for mining frequent itemsets is correct

Note that all itemsets returned by APRIORI are frequent because we return an itemset only if its support is $\geq$ minsup. In other words the algorithm cannot return false positives $\Rightarrow$ to prove correctness we only need to show that no frequent itemset is missed!

Proof of Theorem

By induction on $K \geq 1$ we show that the set F_K computed by APRIORI is the set of *all* frequent itemsets of length K.

BASIS: $K = 1$ Immediate by the specification of the algorithm

INDUCTION STEP: Fix $K > 1$

* Hp. F_{K-1} contains all frequent itemsets of length $K-1$
* We show the F_K = {all frequent itemsets of length K}

Proof of Theorem

$X = X[1]\,X[2]\,\ldots\,X[k]$ arbitrary frequent itemset of length k

Define

$X_1 = X[1]\,X[2]\,\ldots\,X[k-2]\,X[k-1]$ $\quad |X_1| = k-1$

$X_2 = X[1]\,X[2]\,\ldots\,X[k-2]\,X[k]$ $\quad |X_2| = k-1$

E.g. $X = ABCD$ $\quad X_1 = ABC$ $\quad X_2 = ABD$

By anti-monotonicity, $Supp_T(X) \leq Supp_T(X_i)$ $\quad i = 1, 2$

$\Rightarrow$ Since X is frequent then also

X_1, X_2 are frequent $\Rightarrow X_1, X_2 \in F_{k-1}$ by induction hp

Proof of Theorem

$X = X_1 \cup X_2$ and X_1, X_2 share a prefix of length $k-2$ $\Rightarrow$

* X is added to C_k in the generation phase of APRIORI-GEN
* X cannot be removed from C_k in the pruning phase of APRIORI-GEN because all of its subsets of length $k-1$ are frequent and, by inductive hypothesis, they are in F_{k-1}

$\Rightarrow X$ is included in F_k □

Efficiency of A-Priori

A-Priori owes its popularity to its efficient performance especially for sparse datasets (i.e., few frequent itemsets).

The efficiency of A-Priori is due to the following features:

- A few passes over the dataset (typically very large):

  $k_{\max} + 1$ passes, where $k_{\max}$ is the max length of a frequent itemset. Note that for a sparse dataset $k_{\max}$ is small.

- Support is computed only for a few non-frequent itemsets:

  (see lemma in next slide) this is due to candidate generation and pruning which exploit antimonotonicity of support.

- Several optimizations are known for computing candidates supports (typically the most time-consuming task).

Efficiency of A-Priori

Lemma

For dataset T over d items and a threshold minsup, let M be the number of frequent itemsets returned by A-Priori. Then,

The algorithm computes the support for $\leq d + \min\{M^2, dM\}$ itemsets.

Proof of Lemma

Apriori computes support for $d + \sum_{k \geq 2} |C_k|$

- d: individual items
- $\sum_{k \geq 2} |C_k|$: candidates of length ≥ 2

m_k = # of frequent itemsets of length k

$\Rightarrow M = \sum_{k \geq 1} m_k$

For each $Z \in C_k$ there exist X, Y frequent itemsets of length $k-1$ such that

Proof of Lemma

(i) $Z = X \cup Y$

(ii) $Z = X \cup \{a\}$ a last item of Y

this implies

* $|C_k| \leq \binom{m_{k-1}}{2} \leq (m_{k-1})^2$

* $|C_k| \leq m_{k-1} \cdot d$

$\Rightarrow |C_k| \leq \min\{(m_{k-1})^2, d \cdot m_{k-1}\}$

Hence, APRIORI computes supports for

Proof of Lemma

$$d + \sum_{k \geq 2} |c_k|$$

$$\leq d + \min \left\{ \sum_{k \geq 2} (m_{k-1})^2, \sum_{k \geq 2} d \cdot m_{k-1} \right\}$$

$$\leq d + \min \left\{ \left(\sum_{k \geq 2} m_{k-1} \right)^2, d \cdot \sum_{k \geq 2} m_{k-1} \right\}$$

$$\leq d + \min \left\{ M^2, dM \right\}$$

□

Efficiency A-Priori

The following theorem is an easy consequence of the lemma.

Theorem

The A-Priori algorithm for mining frequent itemsets can be implemented in time polynomial in both the input size (sum of all transaction lengths) and the output size (sum of all frequent itemsets legnths).

If the number of candidates for which A-PRIORI computes the support (true by the lemma) the total work will be represented by some polynomial in the input and output size.

A-Priori in practice

The most time-consuming step is the counting of supports of candidates (sets $C_2, C_3, \ldots$)

Main performance issues:

- A-Priori requires a pass over the entire dataset checking each candidate against each transaction (*a lot for large input/output sizes!*)
- The number of candidates may still be very large stressing storage and computing capacity.

  Obs: the issue may become critical for C_2, which contains *all pairs of frequent items* (e.g., suppose 1M items!). As k grows larger, the cardinality of F_{k-1}, hence of C_k, drops.

Many efforts in the last 2 decades to address the above issues!

Other approaches to frequent itemsets mining

Several algorithms based on depth-first mining strategies have been devised and tested (large body of literature!)

Their main features are:

- avoiding several passes over the entire dataset of transactions.
- confining the support counting of longer itemsets to small projections of the dataset.

Prominent examples:

- FP-Growth: proposed in [HJY00] is a popular algorithm for mining frequent itemsets. It is implemented in Spark!
- Patricimine: an improvement to FP-Growth (*devised in Padova!* [PZ03]) is one of the fastest algorithms to date

Approaches to handle large inputs

Suppose that we must compute the frequent itemsets w.r.t. a very large dataset T and a threshold minsup. Two approaches can be used:

1. **Partition-based approach:**
   - Partition T into ℓ subsets $T_1, T_2, \ldots, T_\ell$
   - For each i separately, determine the set F^{T_i} of frequent itemsets w.r.t. T_i and minsup.
   - Compute the support (w.r.t. T) for the itemsets in $\cup_{i=1,\ell} F^{T_i}$, returning those of support $\geq$ minsup.
2. **Sampling approach:** compute the frequent itemsets from a small sample of T.

Remarks:

- The partitioning approach computes the exact output but may be inefficient.
- The sampling approach computes an approximation but is more efficient.

Partition-based approach

The partition-based approach is simple to implement in MapReduce, and the following exercise shows why it is correct.

Exercise

Let T be a dataset of transactions, partitioned into $T_1, T_2, \ldots, T_\ell$. For a given support threshold minsup, define

- F^T = the set of frequent itemsets w.r.t. T,
- F^{T_i} = the set of frequent itemsets w.r.t. T_i,

Show that $F^T \subseteq \bigcup_{i=1}^{\ell} F^{T_i}$.

Remark. The problem with the partition-based approach is that the itemsets which are frequent in the individual partitions (F^{T_i}'s) may be many more than those that are frequent w.r.t. T (F^T), and computing the supports of all of them may be time consuming.

Sampling-based approach

Do we really need to process the entire dataset?

No, if we are happy with some

approximate set of frequent itemsets

(but quality of approximation under control)

Sampling-based approach

Definition (Approximate frequent itemsets)

Consider a dataset T of transactions over the set of items I, a support threshold $\text{minsup} \in (0, 1]$, and a parameter $\epsilon > 0$. A set C of pairs (X, s_X), with $X \subseteq I$ and $s_x \in (0, 1]$, is an ϵ-approximation of the set $F_{T,\text{minsup}}$ of frequent itemsets and their supports if the following conditions are satisfied:

1. For each $X \in F_{T,\text{minsup}}$ there exists a pair $(X, s_X) \in C$
2. For each $(X, s_X) \in C$,
   - $\text{Supp}_T(X) \geq \text{minsup} - \epsilon$
   - $|\text{Supp}_T(X) - s_X| \leq \epsilon$.

→ no frequent itemsets are missed

Similar to the definition of ϵ-Approximate frequent items

Observations

Sampling-based approach: algorithm

Consider a dataset T of N transactions over I and a support threshold $\text{minsup} \in (0, 1]$. The following general algorithm can be used to compute an approximation to the frequent itemsets:

- Fix a suitably lower threshold $f(\text{minsup}) < \text{minsup}$.
- Draw a random sample S from T with uniform probability and with replacement.
- Return the set of pairs

$$C = \left\{ (X, s_x = \text{Supp}_S(X)) \ : \ X \in F_{S,f(\text{minsup})} \right\},$$

where $F_{S,f(\text{minsup})}$ are the frequent itemsets w.r.t. S and minsup.

How well does C approximate the true frequent itemsets and their supports?

Sampling-based approach

Theorem (Riondato-Upfal)

Let h be the maximum transaction length and let ϵ, δ be suitable design parameters in $(0,1)$. There is a constant $c > 0$ such that if

$$f(minsup) = minsup - \frac{\epsilon}{2} \quad AND \quad |S| = \frac{4c}{\epsilon^2}\left(h + \log\frac{1}{\delta}\right)$$

then with probability at least $1-\delta$ the set C returned by the algorithm is an ϵ-approximation of the set of frequent itemsets and their supports.

Terminology: to account for the probabilistic nature of the result in the theorem, the algorithm is said to provide in this case an (ϵ, δ)-approximation to the frequent itemset mining problem.

Observations

Proof of the theorem

The result of the theorem relies on the following lemma which can be proved through an elegant exploitation of the notion of VC-dimension (see [RU14] for details).

Lemma

Using the sample size $|S|$ specified in the theorem, the algorithm ensures that with probability $\geq 1 - \delta$ for each itemset X it holds that

$$|Supp_T(X) - Supp_S(X)| \leq \epsilon/2.$$

Proof of the theorem

Proof of the theorem

Exercises

Exercise

Let T be a dataset of transaction. For an itemset $X = \{x_1, x_2, \ldots, x_k\}$, define the measure:

$$\zeta(X) = \min\{\text{Conf}(x_i \to X - \{x_i\}) \ : \ 1 \leq i \leq k\}.$$

Say whether ζ is *monotone*, *anti-monotone* or neither one. Justify your answer.

Exercise

Let T be a dataset of transactions over a set I of d items, represented by the integers $1, 2, \ldots, d$. For an itemset $X \subseteq I$, let $i_{\max}(X)$ denote the *maximum* item in X (e.g., $i_{\max}(\{2, 6, 7, 11\}) = 11$). For $k \geq 1$, let F_k denote the set of frequent itemsets of length k with respect to a threshold `minsup`. Define

$$C_{k+2} \quad = \quad \{X \cup \{i, j\} : X \in F_k \text{ and } i_{\max}(X) < i < j \leq d\},$$

and show that $F_{k+2} \subseteq C_{k+2}$.

Exercise

Let T be a dataset of N *strings* over an alphabet Σ. Define the support of a string X with respect to T as:

$$\mathrm{Supp}_T(X) = \frac{|\{t \in T \ : \ X \text{ is a substring of } t\}|}{N},$$

that is, the fraction of strings of T that contain X as a substring. (We say that X *is a substring of* t *if it occurs in contiguous positions of* t.)

1. Identify a suitable anti-monotonicity property for $\mathrm{Supp}_T(\cdot)$.
2. Given a support threshold minsup $\in (0, 1)$ let F_k be the set of strings of length k of support $\geq$ minsup. Define

   $$C_{k+1} = \{Xa \ ; \ X \in F_k, a \in \Sigma, \text{ and } X[1] \cdots X[k-1]a \in F_k\}.$$

   Show that $F_{k+1} \subseteq C_{k+1}$.

Summary

- Key notions: itemset; association rule; support of an itemset/association rule); confindence of an association rule.
- Formulation of the problems:
  - mining of frequent itemsets
  - mining of association rules.
- Problematic issues:
  - potential output explosion;
  - false positive/negative.
- Antimonotonicity of support.
- A-Priori algorithm: description; correctness; polynomiality.
- Frequent itemset mining for big data
  - Partition-based approach
  - Sampling-based approach

References

LRU14 J.Leskovec, A.Rajaraman and J.Ullman. Mining Massive Datasets. Cambridge University Press, 2014. Chapter 6.

AS94 R.Agrawal and R.Srikant. Fast algorithms for mining association rules. Proc. 20th International Conference on Very Large Data Bases (VLDB) 1994.

HPY00 J.Han, J.Pei, Y.Yin. Mining frequent patterns without candidate generation. Proc. SIGMOD Conference, 2000.

PZ03 A.Pietracaprina, D.Zandolin: Mining Frequent Itemsets using Patricia Tries. Proc. FIMI 2003.

RU14 M. Riondato, E. Upfal: Efficient Discovery of Association Rules and Frequent Itemsets through Sampling with Tight Performance Guarantees. ACM Trans. on Knowledge Discovery from Data, 2014.