Association Analysis

Part 1

Market-basket model

- 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

Rigorous formulation of the problem

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 $1 \le i \le N$.

Definition (Itemset)

An itemset is a subset $X \subseteq I$ and its support w.r.t. T, denoted by $\mathsf{Supp}_T(X)$, is defined as the *fraction* of transactions of T that contain X.

Definition (Association rule)

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

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

Rigorous formulation of the problem (cont'd)

Given the dataset T of N transcations 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. Compute all frequent itemsets, that is, the set of (non empty) itemsets X such that $\mathsf{Supp}_T(X) \geq \mathsf{minsup}$
2. Compute all (interesting) association rules r such that $\mathsf{Supp}_T(r) \geq \mathsf{minsup}$ and $\mathsf{Conf}_T(r) \geq \mathsf{minconf}$.

Example (minsup=minconf=0.5):

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

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

N.B. For simplicity, the subscript T will be omitted if clear from the context

Observations

- Support and confidence measure the interestingness of a pattern (itemset or rule). In particular, the thresholds *minsup* and *minconf* define which patterns must be regarded as interesting.
- (*Hypothesis testing setting*) Ideally, we would like that the support and confidence (for rules) of the returned patterns be unlikely to be seen in a *random* dataset. However, what is a *random dataset*?
- The choice of minsup and minconf is crucial since it directly influences
  - The output size: low thresholds may yield too many patterns (possibly exponential in the input size) which become hard to exploit effectively.
  - The number of *false positive/negatives*: low thresholds may yield a lot of uninteresting patterns (false positives), while high thresholds may miss some interesting patterns (false negatives).

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.

The mining of frequent itemsets can be generalized to search for: motifs in (biological) sequences or networks; sequential patterns; etc. Techniques devloped for itemsets can be exploited also in these contexts.

Potential output explosion

Let I be a set of d items.

Theorem

The number of distinct non-empty itemsets over I is $2^d - 1$, while the number of distinct association rules is $3^d - 2^{d+1} + 1$.

- Strategies that enumerate of all itemsets/rules in order to find the interesting ones are out of question even for ground sets I of small size (say $d > 40$)
- As a first approximation, we consider efficient strategies those that require time/space polynomial in both the input and the output sizes. (Polynomiality only w.r.t. input may not be feasible if output is large!)

Potential output explosion (cont'd)

Proof of theorem.

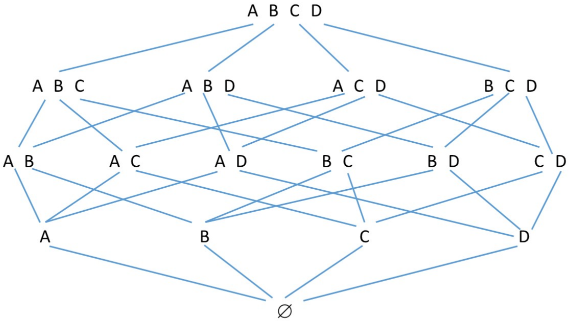
The count of itemsets is trivial. As for the association rules, we count separately those whose LHS has k items, for $1 \leq k < d$. There are $\binom{d}{k}$ possible itemsets of size k, and each of these, say X, can form a rule with $2^{d-k} - 1$ distinct non-empty itemsets, disjoint from X. Thus, the total number of rules is:

$$\begin{aligned}
\sum_{k=1}^{d-1} \binom{d}{k} (2^{d-k} - 1) &= \sum_{k=0}^{d-1} \binom{d}{k} (2^{d-k} - 1) - (2^d - 1) \\
&= \left(\sum_{k=0}^{d} \binom{d}{k} 2^{d-k} \right) - \left(\sum_{k=0}^{d} \binom{d}{k} \right) - (2^d - 1) \\
&= 3^d - 2^d - 2^d + 1 = 3^d - 2^{d+1} + 1
\end{aligned}$$

Use the fact $\sum_{k=0}^{d} \binom{d}{k} 2^{d-k} = \sum_{k=0}^{d} \binom{d}{k} 1^k 2^{d-k}$. $\square$

Lattice of Itemsets

- The family of itemsets under $\subseteq$ forms a lattice, namely a poset where for each to elements X, Y there exists a 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

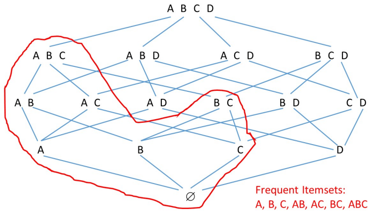
The support function for itemsets exhibits the following property, referred to as anti-monotonicity. For every $X, Y \subseteq I$

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

Immediate consequence. For a given support threshold, we have

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

This implies that, in the lattice, frequent itemsets form a sublattice closed downwards



Efficient mining of F.I. and A.R.

Key objectives:

- Careful exploration of the lattice of itemsets exploiting anti-monotonicity of support
- Time/space complexities polynomial in the input and output size.

Two phases: (typical of most algorithms)

Phase 1: Compute the set F of all frequent itemsets w.r.t. minsup

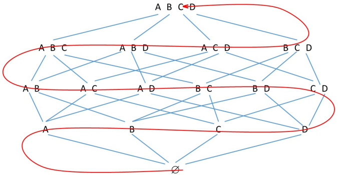
Phase 2: For each itemset $Z \in F$, compute all rules $r : X \rightarrow Y$, with $X \cup Y = Z$ and confidence at least minconf.

Observation. Phase 1 is, usually, the most demanding, computationally.

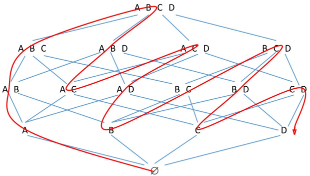
Efficient mining of F.I. and A.R. (cont'd)

Two main approaches

Breadth First



Depth First



F.I. mining: A-Priori algorithm

- A-Priori is a popular, paradigmatic data mining algorithm devised by Agrawal and Srikant in 1994 at IBM Almaden (presented at 20th Very Large Data Base Conf., 1994).
- Uses the breadth-first approach
- Executes $k_{\max} + 1$ passes over the dataset, where $k_{\max}$ is the length of the longest frequent itemset

F.I. mining: A-Priori algorithm (cont'd)

For every itemset $X \subseteq I$, define its absolute support (with respect to a dataset T of N transactions)

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

MAIN ALGORITHM

Input Dataset T of N transactions over I, minsup

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

F.I. mining: A-Priori algorithm (cont'd)

$k \leftarrow 1$
Compute $F_1 = \{i \in I \ ; \ \texttt{Supp}(\{i\}) \geq \texttt{minsup}\}$;
Compute $O_1 = \{(X, \texttt{Supp}(X)) \ ; \ X \in F_1\}$
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**
    $C_{k,t} \leftarrow \{c \in C_k \ : \ c \subseteq t\}$
    for each $c \in C_{k,t}$ **do** $\sigma(c) \leftarrow \sigma(c) + 1$
  $F_k \leftarrow \{c \in C_k \ : \ \sigma(c) \geq N \cdot \texttt{minsup}\}$;
  $O_k \leftarrow \{(X, \sigma(X)/N) \ ; \ X \in F_k\}$
until $F_k = \emptyset$
return $\bigcup_{k \geq 1} O_k$

F.I. mining: A-Priori algorithm (cont'd)

APRIORI-GEN(F)

```
Let ℓ−1 be the size of each itemset in F
Φ ← ∅
/* Candidate Generation */
for each X, Y ∈ F s.t. X ≠ Y ∧ X[1...ℓ−2] = Y[1...ℓ−2] do
  add X ∪ Y to Φ
/* Candidate Pruning */
for each Z ∈ Φ do
  for each Y ⊂ Z s.t. |Y| = ℓ−1 do
    if (Y ∉ F) then {remove Z from Φ; break}
return Φ
```

Observation: no itemset is generated twice

Example

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

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

Example (cont'd)

Set C_2			Set F_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	*

Example (cont'd)

Set C_3			Set F_3
ITEMSET		SUPPORT	
After Generation	After Pruning		
ABC			
ABF	ABF	1/2	*
ACF			
BEF	BEF	1/2	*

Correctness of A-Priori

Assume existence of a total ordering of the items, and consider transactions/itemsets as sorted vectors.

Theorem (Correctness)

The A-Priori algorithm for mining frequent itemsets is correct

Proof

By induction on $k \geq 1$, we show that the set F_k computed by the algorithm consists of all frequent itemsets of size k.

- Basis $k = 1$: trivial
- Induction step. Fix $k > 1$ and assume (inductive hypothesis) the property holds up to index $k - 1$. It is sufficient to prove that for an arbitrary frequent itemset X of size k, X is surely included in the set C_k returned by APRIORI-GEN(F_{k-1}).

Correctness of A-Priori (cont'd)

Proof (cont'd).

Let $X = x_1 x_2 \cdots x_k$ and define

$$\begin{aligned} X(1) &= x_1 x_2 \cdots x_{k-2} x_{k-1} \\ X(2) &= x_1 x_2 \cdots x_{k-2} x_k. \end{aligned}$$

Clearly, $X = X(1) \cup X(2)$. Also, both $X(1)$ and $X(2)$ have length $k-1$ and are frequent, by anti-monotonicity of support. Thus, $X(1), X(2) \in F_{k-1}$, hence X is added to the pool of candidates in the generation phase and cannot be eliminated in the pruning phase, since, being frequent, all of its subsets are also frequent.

□

Complexity of A-Priori

The most delicate step is the support computation for all candidates returned by APRIORI-GEN: in particular, the instruction

$$C_{k,t} \leftarrow \{c \in C_k \ : \ c \subseteq t\}$$

for a given transaction t.

Let n_k be the number of candidates in C_k and n_t the length of t. Assume that C_k is stored in a suitable data structure such as a *trie* (see next slide)

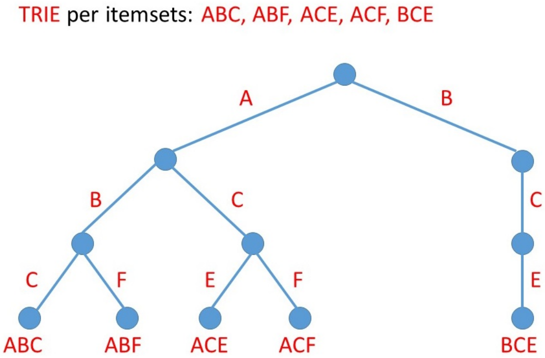
- If $\binom{n_t}{k} \leq n_k$ then for generate all subsets of t of size k and add to $C_{k,t}$ those that belong to C_k.
- If $\binom{n_t}{k} > n_k$ then scan all candidates in C_k and add to $C_{k,t}$ those that are subsets of t.

In any case, the support computation for all candidates can be accomplished in $O\left(n_k \sum_{t \in T} n_t\right)$

TRIE

A TRIE (word derives from RETRIEVAL), also called *prefix tree*, is a kind of search tree used to store a (dynamic) set of strings. Edges are labelled with characters, and the strings are associated with the leaves, where for each leaf v the associated string is given by the characters encountered in the path from the root to v.

Example:



Complexity of A-Priori (cont'd)

Consider a dataset T of N transactions over a set I of d items, and a support threshold minsup. Define

- W_{in}: sum of transactions lengths (input size)
- W_{out}: sum of lengths of frequent itemsets (output size)
- $k_{\max}$: length of longest frequent itemset

Theorem

The A-Priori algorithm for mining frequent itemsets runs in time polynomial in W_{in} and W_{out}.

Proof

Let us determine the contribution of the main steps of the algorithm to the complexity.

- Computation of F_1 and O_1: time $O\left(d \cdot W_{\mathrm{in}}\right)$. Since $d \leq W_{\mathrm{in}}$, the time is $O\left(W_{\mathrm{in}}^2\right)$.

Complexity of A-Priori (cont'd)

Proof (cont'd)

- APRIORI-GEN(F_{k-1}), for $2 \leq k \leq k_{\max} + 1$:
  - Candidate generation: time $O\left(k|F_{k-1}|^2\right)$, where $|F_{k-1}|$ denotes the numebr of frequent itemsets of size $k-1$.
  - Candidate pruning: time $O\left(k^2|F_{k-1}|^2\right)$, assuming F_{k-1} stored in a trie.

  Overall, all executions of APRIORI-GEN require time $O\left(\sum_{k=2}^{k_{\max}+1}(k^2|F_{k-1}|^2)\right)$, hence, time $O\left(W_{\mathrm{out}}^2\right)$.
- Support counting for C_k, for $2 \leq k \leq k_{\max} + 1$: time $O\left(n_k W_{\mathrm{in}}\right)$, where n_k denotes the number of candidates of size k. Since $n_k < |F_{k-1}|^2$, we have that, overall, all support countings require time $O\left(\sum_{k=2}^{k_{\max}+1}(|F_{k-1}|^2 W_{\mathrm{in}})\right)$, hence, time $O\left(W_{\mathrm{in}} W_{\mathrm{out}}^2\right)$.

Optimizations of A-Priori: frequent pairs

- The support counting for the candidates in C_k, for $k \geq 2$, is typically the most time-consuming step because: (a) requires a pass over the entire dataset; (b) may use much space (number of candidates can be quadratic in the actual number of frequent itemsets)
- In practice, the issue of space (point (b) above) may become critical for C_2, which contains *all pairs of frequent items*. As k grows larger, the cardinality of F_{k-1}, hence of C_k, drops.
- Park, Chen and Yu [SIGMOD'95] devised a strategy to filter out some candidates from C_2 based on statistics gathered while computing F_1. This strategy is outlined in the next slide.

Optimizations of A-Priori: frequent pairs (cont'd)

GOAL: Compute F_1 efficiently and, at the same time, gather statistics for filtering out infrequent pairs. Consider an instance with N transactions, d items, and threshold minsup

- Let h be a hash function that maps pairs of items to integers in $[0, K-1]$, for a suitable value K
- Use $d + K$ counters: one counter γ_i for each $i \in I$, and a counter δ_j, for every $0 \leq j < K$. Counters are initialized to 0.
- For every transaction t do
  - For each each item $i \in t$, increment γ_i
  - For each pair of items $i_1, i_2 \in t$, increment $\delta_{h(i_1,i_2)}$.
- Key remark: only pairs of items i_1, i_2 such that $\delta_{h(i_1,i_2)} \geq N \cdot \text{minsup}$ have a chance to be frequent.

Optimizations of A-Priori: frequent pairs (cont'd)

- Compute $F_1 = \{i \in I \ : \ \gamma_i \geq N \cdot \text{minsup}\}$
- Compute C_2 as the set of pairs i_1, i_2 such that:

$$(i_1, i_2 \in F_1) \quad \text{AND} \quad (\delta_{h(i_1,i_2)} \geq N \cdot \text{minsup})$$

Observations:

- The first condition yields the same set of candidates as APRIORI-GEN, while the second condition aims at filtering out some of these candidates.
- If K is chosen sufficiently large (based on the available memory), hence many pair counters are used, then filtering out of infrequent pairs become quite effective.

Other optimizations

A large body of literature has investigated several additional strategies to optimize the mining of frequent itemsets. E.g.:

- A data structure trie-like (Hash tree) was defined by the original developers of A-Priori, to be used for storing the set of candidates C_k so to speed up their support counting. In essence, for each transaction, the hash tree quickly provides a subset of the candidates (smaller than C_k) to be checked for inclusion.
- Several implementations of depth-first mining strategies have been devised and tested (one of the fastest ones to date, *patriciamine*, come from Padova!). Their goal is to avoid several passes over the entire dataset of transactions, which may be huge, and to confine the support counting of longer itemsets to suitable projections of the dataset, typically much smaller than the original one.

Mining association rules

- Once the frequent itemsets and their supports have been computed (set $\bigcup_{k \geq 1} O_k$ returned by A-Priori) all association rules which are based on these itemsets and satisfy the given confidence requirement can be determined.
- Let minconf be the given confidence threshold. For each frequent itemset Z, we must determine the set:

$$\{r \; : \; Z - Y \rightarrow Y \;\; \text{s.t.} \;\; \emptyset \neq Y \subset Z \; \wedge \; \mathrm{Conf}(r) \geq \mathrm{minconf}\}$$

- Note that each rule in the above set has the same support as Z, hence it automatically satisfies the support constraint since Z is frequent. Conversely, rules derived from itemsets which are not frequent need not to be checked, since they would not satisfy the support constraint .

Mining association rules (cont'd)

- Checking all non-empty subsets $Y \subseteq Z$ as RHS of possible rules with confidence at least minconf may be too costly. We exploit a sort of anti-monotonicity property for rules, as we did for frequent itemsets.
- Anti-monotonicity property for rules. For $\emptyset \neq Y' \subset Y \subset Z$, we have:

$$\frac{\mathrm{Supp}(Z)}{\mathrm{Supp}(Z-Y)} \leq \frac{\mathrm{Supp}(Z)}{\mathrm{Supp}(Z-Y')}$$

- Immediate consequence.

$$\mathrm{Conf}(Z - Y' \to Y') < \mathrm{minconf} \Rightarrow \mathrm{Conf}(Z - Y \to Y) < \mathrm{minconf}.$$

  Thus, for each frequent itemset Z it is convenient to check rules with RHS of progressively increasing size.

Algorithm for mining association rules

Let $O =$ set of frequent itemset and their supports

ASSOCIATION RULE ALGORITHM

Input O, $\texttt{minconf}$
Output $\{(r, \texttt{Supp}(r), \texttt{Conf}(r)) \; : \; \texttt{Supp}(r) \geq \texttt{minsup} \wedge \texttt{Conf}(r) \geq \texttt{minconf}\}$

$R \leftarrow \emptyset$
for each Z $\texttt{s.t.}$ $|Z| > 1 \wedge (Z, \texttt{support}(Z)) \in O$ **do**
  $R \leftarrow R \cup$ AP-GENRULES(Z)
return R

Algorithm for mining association rules (cont'd)

AP-GENRULES(Z)

$m \leftarrow 1$
$H_{Z,1} \leftarrow \{Y \subset Z \ : \ |Y| = 1 \wedge \mathtt{Supp}(Z)/\mathtt{Supp}(Z - Y) \geq \mathtt{minconf}\}$
$R_{Z,1} \leftarrow \{(r, \mathtt{Supp}(r), \mathtt{Conf}(r)) \ : \ r : Z - Y \rightarrow Y, \ \mathtt{with} \ Y \in H_{Z,1}\}$
repeat
  if $(m + 1 = |Z|)$ **then break**
  $m \leftarrow m + 1$
  $H_{Z,m} \leftarrow$ APRIORI-GEN$(H_{Z,m-1})$
  $R_{Z,m} \leftarrow \emptyset$
  for each $Y \in H_{Z,m}$ **do**
    if $(\mathtt{Supp}(Z)/\mathtt{Supp}(Z - Y) \geq \mathtt{minconf})$
      then `add` $(r : Z - Y \rightarrow Y, \mathtt{Supp}(r), \mathtt{Conf}(r))$ `to` $R_{Z,m}$
      else `remove` Y `from` $H_{Z,m}$
until $H_{Z,m} = \emptyset$
return $\bigcup_{m \geq 1} R_{Z,m}$

Example

Set O	
ITEMSET	SUPPORT
A	$1/2$
B	$3/4$
C	$3/4$
E	$3/4$
AC	$1/2$
BC	$1/2$
BE	$3/4$
CE	$1/2$
BCE	$1/2$

Consider AP-GENRULES(Z), with $Z = BCE$ and minconf $= 3/4$. We have

$$
\begin{aligned}
H_{Z,1} &= \{E, B\} \\
R_{Z,1} &= \{(BC \rightarrow E, 1/2, 1), \\
&= (CE \rightarrow B, 1/2, 1)\}
\end{aligned}
$$

Note that $C \notin H_{Z,1}$ since $\text{Conf}(BE \rightarrow C) = 2/3 <$ minconf.

In the first iteration of the repeat, the algorithm computes $H_{Z,2} = \{BE\}$ thorugh APRIORI-GEN, but then it removes BE from $H_{Z,2}$ since $\text{Conf}(C \rightarrow BE) = 2/3 <$ minconf. Thus, both $H_{Z,2}$ and $R_{Z,2}$ are empty at the end of the iteration, and the rules $BC \rightarrow E$ and $CE \rightarrow B$ are returned (with their supports and confidences).

Correctness of A.R. algorithm

Theorem (Correctness)

The algorithm for mining association rules is correct

Proof

Let Z be a frequent itemset of size $k > 1$. By induction on $m \geq 1$ we show that the set $H_{Z,m}$ computed in each iteration of AP-GENRULES (first initialized by APRIORI-GEN$(H_{Z,m-1})$ and then suitably pruned in the for-each loop) coincides with the set of consequents $Y \subset Z$ of size m such that $\mathrm{Conf}(Z - Y \rightarrow Y) \geq$ minconf.

Correctness of A.R. algorithm (cont'd)

Proof (cont'd).

- Basis $m = 1$: trivial
- Induction step. Let us fix $m > 1$ and assume (inductive hypothesis) that the property holds up to index $m - 1$. In particular, we assume inductively, that $H_{Z,m-1}$ coincides with the set of consequents $Y \subset Z$ of size $m - 1$ such that the rule $r : Z - Y \rightarrow Y$ has confidence at least minconf. It is sufficient to prove that given an arbitrary subset $Y \subset Z$ of size m such that the rule $r : Z - Y \rightarrow Y$ has confidence at least `minconf`, Y is included in the output of APRIORI-GEN$(H_{Z,m-1})$. The argument is similar to the one used to prove correctness of the frequent itemsets mining algorithm and is left as an exercise.

□

Observations

1. The algorithm for extracting association rules from the frequent itemsets does not require access to the dataset T but only to the frequent itemsets and their supports. If the frequent itemsets are not too many, as one would hope when the support threshold is properly chosen, avoiding the access to T may yield substantial performance gains.
2. Let U_{in} denote the sum of the lengths of the frequent itemsets, and U_{out} the sum of the lengths of the returned association rules (i.e., the length of the union of LHS and RHS), arguments similar to the one developed for the analysis of A-Priori can show that the time required by the algorithm for extracting association rules from the frequent itemsets is polynomial in U_{in} and U_{out}.

Exercises

Exercise 1

Argue rigorously that given a family F of itemsets of the same length, represented as sorted arrays of items, function $\textsc{apriori-gen}(F)$ does not generate the same itemset twice.

Exercise 2

Consider a dataset T of transactions over a set I of d items and suppose that there exist M frequent itemsets w.r.t. some support threshold minsup. Show that A-Priori explicitly computes the support of at most $d + \min\{M^2, dM\}$ itemsets.

Exercise 3

Consider two association rules $r_1 : A \rightarrow B$, and $r_2 : B \rightarrow C$, and suppose that both satisfy support and confidence requirements. Is it true that also $r_3 : A \rightarrow C$ satisfies the requirements? If so, prove it, otherwise show a counterexample.

Exercises

Exercise 4

Let

$$\begin{aligned} c_1 &= \text{Conf}(A \to B) \\ c_2 &= \text{Conf}(A \to BC) \\ c_3 &= \text{Conf}(AC \to B) \end{aligned}$$

What relationships do exist among the $c_i's$?

Exercise 5

For a given 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.

Exercises

Exercise 6

Consider the following alternative implementation of procedure $\textsc{Apriori-Gen}(F_{k-1})$ (regard an itemset $X \in F_{k-1}$ as an array of items $X[1], X[2], \ldots, X[k-1]$ in increasing order):

$C_k \leftarrow \emptyset$;
for each $X \in F_{k-1}$ **do**
  for each `(`$i \in F_1$`)` **do**
    if `(`$i > X[k-1]$`)` **then** `add` $X \cup \{i\}$ `to` C_k
`remove from` C_k `every itemset containing at least`
  `one subset of length` $k-1$ `not in` F_{k-1}
return C_k

Show that the set C_k returned by the above procedure contains all frequent itemsets of length k.

References

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

TSK06 P.N.Tan, M.Steinbach, V.Kumar. Introduction to Data Mining. Addison Wesley, 2006. Chapter 6.