Approximate counting: count-min data structure

G. Cormode and S. Muthukrishhan: An improved data stream summary: the count-min sketch and its applications. Journal of Algorithms 55 (2005) 58-75.

- Problem definition
- 2-independent hash functions
- Count-min data structures

Problem definition

- A large set U of potential identifiers (e.g., IP addresses or sentences)
- Keep counts associated with each $u \in U$
- For $u \in U$ and integer c operations $\texttt{inc}(u, b)$ and $\texttt{dec}(u, b)$: increase or decrease the count associated with u by b; assume the counts stay nonnegative
- Queries: approximate counts $C(u)$: what is the count associated to u?
- Initially, $C(u) = 0$ for all $u \in U$; $S = \sum_{u \in U} C(u)$
- Heavy hitters: which items u satisfy $C(u) \geq \alpha S$ (approximately)
- U is so large that we cannot keep an array element for each u seen in the query stream

Basic idea

- Keep several hash tables A_i, $i = 1, \ldots, k$ and hash functions h_i
- Implement `inc`(u, c) by doing $A_i[h_i(u)] = A_i[h_i(u)] + c$ and `dec`(u, c) by doing $A_i[h_i(u)] = A_i[h_i(u)] - c$
- Answer question $C(u)$? by

$$min_{i=1}^{k} A_i[h_i(u)]$$

- Heavy hitters: use these counts to keep a heap structure

Families of universal hash functions

- U the universe with $|U| \geq n$, and $V = \{0, 1, \ldots, n-1\}$
- $\mathcal{H}$: a family of functions $h : U \to V$
- $\mathcal{H}$ is 2-universal, if for any $u_1, u_2 \in U$ and for a uniformly selected function $h \in \mathcal{H}$ we have

$$Pr(h(u_1) = h(u_2)) \leq \frac{1}{n}.$$

- Thus collisions are about as rare as they can be
- Note that h is selected at random, and that the claim holds for all u_1, u_2
- Also called pairwise independent hash functions

A simple universal family

- Assume $U = \{0, \ldots, m-1\}$ and $V = \{0, 1, \ldots, n-1\}$
- Let $p \geq m$ be prime
- $h_{a,b}(u) = ((ax + b) \bmod p) \bmod n$
- Family

$$\mathcal{H} = \{h_{a,b} | 1 \leq a \leq p-1, 0 \leq b \leq p\}$$

is 2-universal.

Count-min data structure

- Parameters ε (the accuracy we want to have) and δ (the certainty with which we reach the accuracy)
- $w = \lceil e/\varepsilon \rceil$ (here e is the base of ln)
- $d = \lceil \ln(1/\delta) \rceil$
- Array A of size $d \times w$, initially 0
- Pairwise independent hash functions $h_1, \ldots, h_d : \{0, \ldots, m-1\} \to \{1, \ldots, w\}$

Update procedure and answering count queries

- `inc`(u, c) by doing $A[i, h_i(u)] = A[i, h_i(u)] + c$ for all $i = 1, \ldots, d$
- `dec`(u, c) $A[i, h_i(u)] = A[i, h_i(u)] - c$ for all $i = 1, \ldots, d$
- Answer $C(u)$ by returning $\hat{c} = \min_j A[j, h_j(u)]$

Example

ε	δ	w	d	wd
0.1	0.1	28	3	84
0.1	0.01	28	5	140
0.1	0.001	28	7	196
0.01	0.1	272	3	816
0.01	0.01	272	5	1360
0.01	0.001	272	7	1904
0.001	0.001	2719	7	19033

Properties

- Estimates are never too small: $C(u) \leq \hat{c}$
- With probability at least $1 - \delta$

$$\hat{c} \leq C(u) + \varepsilon S$$

- What does this mean? If S is small, the estimates are accurate, if S is large, only counts that are large are estimated accurately.
- Recall $S = \sum_{u \in U} C(u)$

Proof

- Indicator variables $I(u,j,v)$: equal to 1 if and only if $u \neq v$ and $h_j(u) = h_j(v)$, 0 otherwise
- Pairwise independence:

$$E(I(u,j,v) = Pr(h_j(u) = h_j(v)) \leq 1/w = \varepsilon/e.$$

- $X(u,j) = \sum_{v \in U} I(u,j,v)C(v)$
- $A[j, h_j(u)] = C(u) + X(u,j)$
- Thus $C(u) \leq \hat{c} = \min_j A[j, h_j(u)]$

$$E(X(u,j) = E(\sum_{v \in U} I(u,j,v)C(v)) \leq \sum_{v \in U} C(v)E(I(u,j,v)) \leq S\frac{\varepsilon}{e}$$

Markov's inequality (but is pairwise independence enough?)

$$\begin{aligned}
Pr(\hat{c} > C(u) + \varepsilon S &= Pr(\forall j : A[j, h_j(u)] > C(u) + \varepsilon S) \\
&= Pr(\forall j : C(u) + X(u,j) > C(u) + \varepsilon S) \\
&= Pr(\forall j : X(u,j) > \varepsilon S) \\
&\leq Pr(\forall j : X(u,j) > eE(X(u,j))) \\
&= \bigcap_{j=1}^{d} Pr(X(u,j) > eE(X(u,j))) \\
&< (\frac{1}{e})^d = e^{-d} \leq \delta
\end{aligned}$$

Answering heavy hitter queries

- Maintain S all the time
- When seeing `inc` operations, see if the frequency seems to be high enough
- Decreasing the counts makes things more difficult

Chapter 3: Frequent sets and association rules

Chapter 3. Frequent sets and association rules

How to count frequently occurring subsets in data?

- 1. Problem formulation
- 2. Rules from frequent sets
- 3. Finding frequent sets
- 4. Experimental results
- 5. Related issues
- 6. Rule selection and presentation
- 7. Theoretical results

Example

- Customer 1: mustard, sausage, beer, chips
  Customer 2: sausage, ketchup
  Customer 3: beer, chips, cigarettes
  ...
  Customer 236513: coke, chips
- A set of products $X\{$ mustard, chips $\}$
- Frequency $f(X)$ of X in the dataset: the fraction of rows that have all elements of X
- Basic task: find all sets X such that $f(X) > c$ for a given constant c
- A **frequent set**

Problem formulation: data

- a set R of items
- a *0/1 dataset* r *over* R (or 0-1 relation) is a collection (or multiset) of subsets of R
- the elements of r are called *rows*
- the number of rows in r is denoted by $|r|$
- the *size* of r is denoted by $||r|| = \sum_{t \in r} |t|$

Row ID	Row
t_1	$\{A, B, C, D, G\}$
t_2	$\{A, B, E, F\}$
t_3	$\{B, I, K\}$
t_4	$\{A, B, H\}$
t_5	$\{E, G, J\}$

Figure 1: An example 0/1 relation r over the set $R = \{A, \ldots, K\}$.

Example

Row ID	A	B	C	D	E	F	G	H	I	J	K
t_1	1	1	1	1	0	0	1	0	0	0	0
t_2	1	1	0	0	1	1	0	0	0	0	0
t_3	0	1	0	0	0	0	0	0	1	0	1
t_4	1	1	0	0	0	0	0	1	0	0	0
t_5	0	0	0	0	1	0	1	0	0	1	0

a 0/1 relation over the schema $\{A, \ldots, K\}$

Notation

- Sometime we write just ABC for $\{A, B, C\}$ etc.
- Attributes = variables
- An observation in the data is
  - A set of attributes, or
  - A row of 0s and 1s

Patterns: sets of items

- r a 0/1 relation over R
- $X \subseteq R$
- X *matches* a row $t \in r$, if $X \subseteq t$
- the set of rows in r matched by X is denoted by $\mathcal{M}(X, r)$, i.e., $\mathcal{M}(X, r) = \{t \in r \mid X \subseteq t\}$.
- the *(relative) frequency* of X in r, denoted by *fr*(X, r), is

$$\frac{|\mathcal{M}(X, r)|}{|r|}.$$

- Given a *frequency threshold min_fr* $\in [0, 1]$, the set X is *frequent*, if *fr*$(X, r) \geq$ *min_fr*.

Frequent sets

- given R (a set), r (a 0/1 relation over R), and *min_fr* (a frequency threshold)
- the collection of frequent sets $\mathcal{F}(r, \mathit{min\_fr})$

$$\mathcal{F}(r, \mathit{min\_fr}) = \{X \subseteq R \mid \mathit{fr}(X, r) \geq \mathit{min\_fr}\},$$

- In the example relation:
  $\mathcal{F}(r, 0.3) = \{\emptyset, \{A\}, \{B\}, \{E\}, \{G\}, \{A, B\}\}$

Finding frequent sets

- Task: given R (a set), r (a 0/1 relation over R), and *min_fr* (a frequency threshold), find the collection of frequent sets $\mathcal{F}(r, \mathit{min\_fr})$ and the frequency of each set in this collection.
- Count the number of times combinations of attributes occurs in the data

Why find frequent sets?

- Find all combinations of attributes that occur together
- They might be interesting
- Positive combinations only
- Provides a type of summary of the data

When is the task sensible and feasible?

- If *min_fr* $= 0$, then all subsets of R will be frequent, and hence $\mathcal{F}(r, \textit{min\_fr})$ will have size $2^{|r|}$
- Very large, and not interesting
- If there is a subset X that is frequent, then all its subsets will be frequent (why?)
- Thus if a large set X is frequent, there will be at least $2^{|X|}$ frequent sets
- The task of finding all frequent sets is interesting typically only for reasonably large values of *min_fr*, and for datasets that do not have large subsets that would be very strongly correlated.

Finding frequent sets

- trivial solution (look at all subsets of R) is not feasible
- iterative approach
- first frequent sets of size 1, then of size 2, etc.
- a collection $\mathcal{C}_l$ of candidate sets of size l
- then obtain the collection $\mathcal{F}_l(r)$ of frequent sets by computing the frequencies of the candidates from the database
- minimize the number of candidates?

- monotonicity: assume $X' \subseteq X$
- then $fr(X') \geq fr(X)$
- if X is frequent then X' is also frequent
- Let $X \subseteq R$ be a set. If any of the proper subsets $X' \subset X$ is not frequent then (1) X is not frequent and (2) there is a non-frequent subset $X'' \subset X$ of size $|X| - 1$.

Example

$\mathcal{F}_2(r) = \{\{A, B\}, \{A, C\}, \{A, E\}, \{A, F\}, \{B, C\}, \{B, E\}, \{C, G\}\}$,

- then $\{A, B, C\}$ and $\{A, B, E\}$ are the only possible members of $\mathcal{F}_3(r)$,

Levelwise search

- levelwise search: generate and test
- candidate collection:

$$\mathcal{C}(\mathcal{F}_l(r)) = \{X \subseteq R \| X| = l{+}1 \text{ and } X' \in \mathcal{F}_l(r) \text{ for all } X' \subseteq X, |X'| = l\}.$$

Apriori algorithm for frequent sets

Algorithm

Input: A set R, a 0/1 relation r over R, and a frequency threshold *min_fr*.

Output: The collection $\mathcal{F}(r, \mathit{min\_fr})$ of frequent sets and their frequencies.

Method:

1. $\mathcal{C}_1 := \{\{A\} \mid A \in R\}$;
2. $l := 1$;
3. **while** $\mathcal{C}_l \neq \emptyset$ **do**
4. // Database pass (Algorithm 35):
5. compute $\mathcal{F}_l(r) := \{X \in \mathcal{C}_l \mid \mathit{fr}(X, r) \geq \mathit{min\_fr}\}$;
6. $l := l + 1$;
7. // Candidate generation (Algorithm 32):
8. compute $\mathcal{C}_l := \mathcal{C}(\mathcal{F}_{l-1}(r))$;
9. **for** all l and **for** all $X \in \mathcal{F}_l(r)$ **do** output X and $\mathit{fr}(X, r)$;

Correctness

- reasonably clear
- optimality in a sense?
- For any collection $\mathcal{S}$ of subsets of X of size l, there exists a 0/1 relation r over R and a frequency threshold *min_fr* such that $\mathcal{F}_l(r) = \mathcal{S}$ and $\mathcal{F}_{l+1}(r) = \mathcal{C}(\mathcal{S})$.
- fewer candidates do not suffice

Additional information can change things...

- frequent sets: $\{A, B\}$, $\{A, C\}$, $\{A, D\}$, $\{B, C\}$, and $\{B, D\}$
- candidates: $\{A, B, C\}$ and $\{A, B, D\}$
- what if we know that $\mathit{fr}(\{A, B, C\}) = \mathit{fr}(\{A, B\})$
- can infer $\mathit{fr}(\{A, B, D\}) < \mathit{min\_fr}$
- how?

Candidate generation

- how to generate the collection $\mathcal{C}(\mathcal{F}_l(r))$?
- trivial method: check all subsets
- compute potential candidates as unions $X \cup X'$ of size $l+1$
- here X and X' are frequent sets of size l
- check which are true candidates
- not optimal, but fast
- collections of item sets are stored as arrays, sorted in the lexicographical order

Candidate generation algorithm

Algorithm

Input: A lexicographically sorted array $\mathcal{F}_l(r)$ of frequent sets of size l.

Output: $\mathcal{C}(\mathcal{F}_l(r))$ in lexicographical order.

Method:

1. **for** all $X \in \mathcal{F}_l(r)$ **do**
2.   **for** all $X' \in \mathcal{F}_l(r)$ such that $X < X'$ and X and X' share their $l-1$ lexicographically first items **do**
3.   **for** all $X'' \subset (X \cup X')$ such that $|X''| = l$ **do**
4.   **if** X'' is not in $\mathcal{F}_l(r)$ **then** continue with the next X' at line ;
5.   output $X \cup X'$;

Correctness and running time

Theorem 1 *Algorithm 32 works correctly.*

Theorem 2 *Algorithm 32 can be implemented to run in time* $\mathcal{O}(l^2\ |\mathcal{F}_l(r)|^2 \log |\mathcal{F}_l(r)|)$.

Optimizations

compute many levels of candidates at a single pass

$$\begin{aligned}\mathcal{F}_2(r) &= \{\{A,B\},\{A,C\},\{A,D\},\{A,E\},\\ &\quad \{B,C\},\{B,D\},\{B,G\},\{C,D\},\{F,G\}\}.\end{aligned}$$

$$\begin{aligned}\mathcal{C}(\mathcal{F}_2(r)) &= \{\{A,B,C\},\{A,B,D\},\{A,C,D\},\{B,C,D\}\},\\ \mathcal{C}(\mathcal{C}(\mathcal{F}_2(r))) &= \{\{A,B,C,D\}\}, \text{ and}\\ \mathcal{C}(\mathcal{C}(\mathcal{C}(\mathcal{F}_2(r)))) &= \emptyset.\end{aligned}$$

Database pass

- go through the database once and compute the frequencies of each candidate
- thousands of candidates, millions of rows

Algorithm

Input: R, r over R, a candidate collection $\mathcal{C}_l \supseteq \mathcal{F}_l(r, \mathit{min\_fr})$, and $\mathit{min\_fr}$.

Output: Collection $\mathcal{F}_l(r, \mathit{min\_fr})$ of frequent sets and frequencies.

Method:

// Initialization:
for all $A \in R$ **do** $A.\mathit{is\_contained\_in} := \emptyset$;
for all $X \in \mathcal{C}_l$ and **for** all $A \in X$ **do**
    $A.\mathit{is\_contained\_in} := A.\mathit{is\_contained\_in} \cup \{X\}$;
for all $X \in \mathcal{C}_l$ **do** $X.\mathit{freq\_count} := 0$;
// Database access:
for all $t \in r$ **do**
    for all $X \in \mathcal{C}_l$ **do** $X.\mathit{item\_count} := 0$;
    for all $A \in t$ **do**
        for all $X \in A.\mathit{is\_contained\_in}$ **do**
            $X.\mathit{item\_count} := X.\mathit{item\_count} + 1$;
            if $X.\mathit{item\_count} = l$ **then** $X.\mathit{freq\_count} := X.\mathit{freq\_count} + 1$;
// Output:
for all $X \in \mathcal{C}_l$ **do**
    if $X.\mathit{freq\_count}/|r| \geq \mathit{min\_fr}$ **then** output X and $X.\mathit{freq\_count}/|r|$;

Data structures

- for each $A \in R$ a list $A.$*is_contained_in* of candidates that contain A
- For each candidate X we maintain two counters:
  - $X.$*freq_count* the number of rows that X matches,
  - $X.$*item_count* the number of items of X

Correctness

- clear (?)

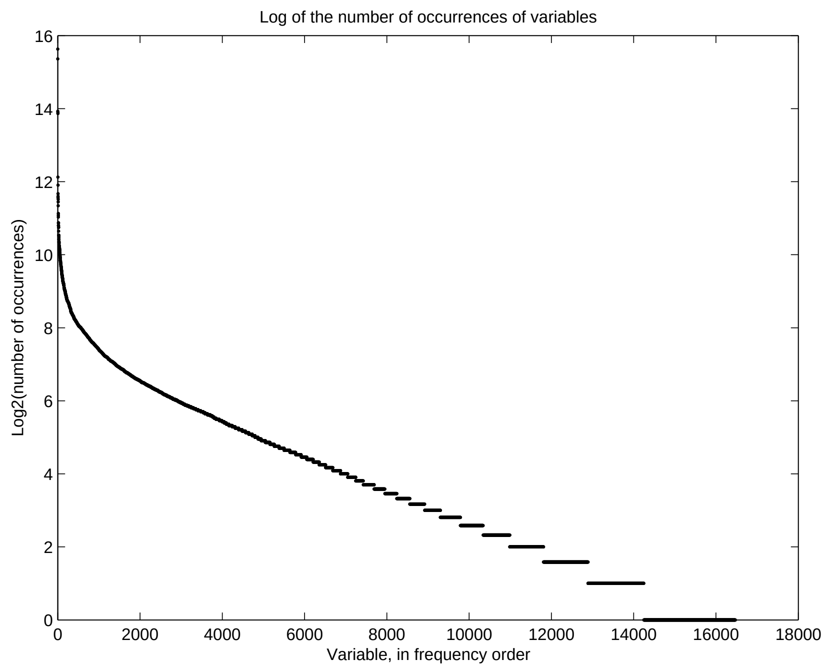
Time complexity

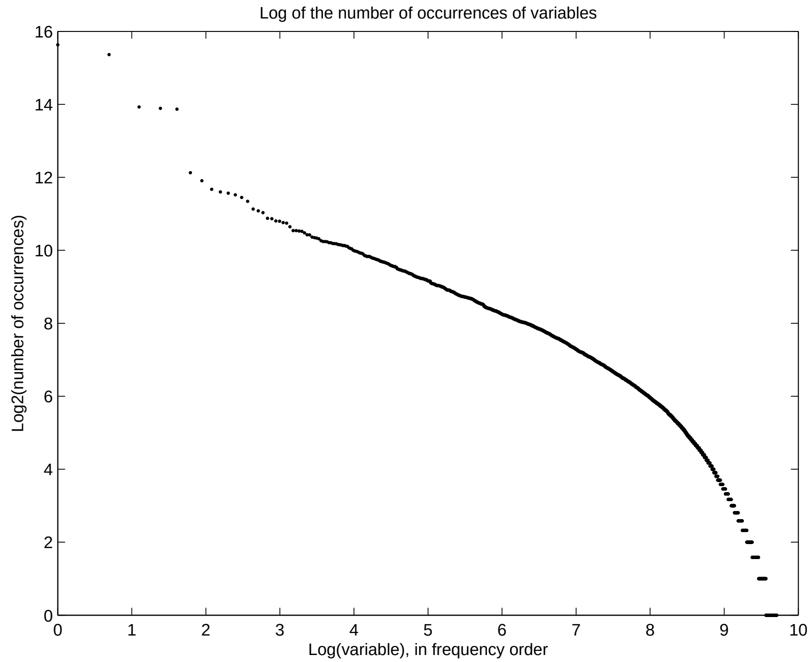
- $\mathcal{O}(||r|| + l\,|r|\,|\mathcal{C}_l| + |R|)$

Implementations

- Lots of them around
- See, e.g., the web page of Bart Goethals
- Typical input format: each row lists the numbers of attributes that are equal to 1 for that row

Experimental results

- A retail transaction dataset, 88162 rows and 16470 columns, 908576 ones
- Rows: transactions, columns: products; completely anonymized
- 


Log of the number of occurrences of variables
Log2(number of occurrences)
Log(variable), in frequency order
0
2
4
6
8
10
12
14
16
0
1
2
3
4
5
6
7
8
9
10

Number of frequent sets

- Number of frequent sets found for different thresholds

Threshold	Sets	Time
5000	15	0.97s
2000	45	1.28s
1000	135	1.65s
500	468	1.92s
400	699	2.14s
300	1135	2.51s
200	2191	3.87s
100	6451	6.83s

Candidates and frequent sets

Threshold	5000		2000		1000		500	
Set size	C	F	C	F	C	F	C	F
1	16470	5	16470	16	16470	56	16470	185
2	10	7	120	20	1540	49	17020	191
3	3	3	14	9	31	24	153	79
4	0	0	2	0	6	6	17	13
5	0	0	0	0	0	0	1	0
6	0	0	0	0	0	0	0	0

Threshold	400		300		200		100	
Set size	C	F	C	F	C	F	C	F
1	16470	274	16470	418	16470	807	16470	1857
2	37401	284	87153	471	325221	895	1723296	2785
3	239	117	413	206	867	411	3430	1475
4	27	22	48	36	110	72	482	306
5	2	2	4	4	6	6	33	28
6	0	0	0	0	0	0	0	0

What do the frequent sets tell us?

- There are lots of combinations of variables (columns) that occur fairly often
- Example: there are 448 rows in which all variables 32 38 39 41 48 are 1
- Is this interesting? Perhaps, if the products are of interest

Is the frequent set statistically significant?

- Focus on the set 32 38 39 41 48; how likely it is to see 448 occurrences of this set in 88162 rows, if the variables were independent?
- Frequencies of the individual variables:
  32 15167 0.172036
  38 15596 0.176902
  39 50675 0.574794
  41 14945 0.169517
  48 42135 0.477927
- Probability of having all 5: 0.00141722; implies on expectation 125 occurrences
- Chernoff bound: the probability of seeing 448 occurrences in 88162 tries with probability 0.00141722 is very low

But?

- We searched a lot of potential frequent sets
- Would a similar frequent set occur if the data were random?
- Would a similar number of frequent sets occur if the data were random?
- We will return to this issues later

Association rules

- Let R be a set, r a 0/1 relation over R, and $X, X' \subseteq R$ sets of items
- $X \Rightarrow X'$ is an *association rule* over r.
- The *accuracy* of $X \Rightarrow X'$ in r, denoted by *conf*$(X \Rightarrow X', r)$, is

$$\frac{|\mathcal{M}(X \cup X', r)|}{|\mathcal{M}(X, r)|}.$$

- *conf*$(X \Rightarrow X', r)$: the conditional probability that a row in r matches X' given that it matches X

Association rules II

- The *frequency* *fr*$(X \Rightarrow X', r)$ of $X \Rightarrow X'$ in r is *fr*$(X \cup X', r)$.
  - frequency is also *called support*
- a *frequency threshold min_fr* and a *accuracy threshold min_conf*
- $X \Rightarrow X'$ *holds* in r if and only if *fr*$(X \Rightarrow X', r) \geq$ *min_fr* and *conf*$(X \Rightarrow X', r) \geq$ *min_conf*.

Discovery task

- given R, r, *min_fr*, and *min_conf*
- find all association rules $X \Rightarrow X'$ that hold in r with respect to *min_fr* and *min_conf*
- X and X' are disjoint and non-empty

- *min_fr* $= 0.3$, *min_conf* $= 0.9$
- The only association rule with disjoint and non-empty left and right-hand sides that holds in the database is $\{A\} \Rightarrow \{B\}$
- frequency 0.6, accuracy 1
- when is the task feasible? interesting?
- note: asymmetry between 0 and 1

How to find association rules

- Find all frequent item sets $X \subseteq R$ and their frequencies.
- Then test separately for all $X' \subset X$ with $X' \neq \emptyset$ whether the rule $X \setminus X' \Rightarrow X'$ holds with sufficient accuracy.
- Latter task is easy.
- exercise: rule discovery and finding frequent sets are equivalent problems

Rule generation

Algorithm

Input: A set R, a 0/1 relation r over R, a frequency threshold *min_fr*, and a accuracy threshold *min_conf*.

Output: The association rules that hold in r with respect to *min_fr* and *min_conf*, and their frequencies and accuracies.

Method:
1. // Find frequent sets (Algorithm 28):
2. compute $\mathcal{F}(r, \mathit{min\_fr}) := \{X \subseteq R \mid \mathit{fr}(X, r) \geq \mathit{min\_fr}\}$;
3. // Generate rules:
4. **for** all $X \in \mathcal{F}(r, \mathit{min\_fr})$ **do**
5.   **for** all $X' \subset X$ with $X' \neq \emptyset$ **do**
6.   **if** $\mathit{fr}(X)/\mathit{fr}(X \setminus X') \geq \mathit{min\_conf}$ **then**
7.   output the rule $X \setminus X' \Rightarrow X'$, $\mathit{fr}(X)$, and $\mathit{fr}(X)/\mathit{fr}(X \setminus X')$;

Correctness and running time

- the algorithm is correct
- running time?

Examples

- Same data as before
- Accuracy threshold 0.9
-

Frequency	Rules
5000	0
2000	4
1000	14
500	32
400	44
300	64
200	100
100	220

Example rules

36 39 41 =¿ 38 (553, 0.966783)

36 39 48 =¿ 38 (1080, 0.967742)

36 41 =¿ 38 (671, 0.958571)

36 48 =¿ 38 (1360, 0.960452)

37 =¿ 38 (1046, 0.973929)

37 39 =¿ 38 (684, 0.967468)

37 48 =¿ 38 (557, 0.985841)

Are these interesting?

Rule selection and presentation

- Recall the KDD process
- association rules etc.: idea is to generate **all** rules of a given form
- Lots of rules
- All rules won't be interesting
- How to make it possible for the user to find the truly interesting rules? Second-order knowledge discovery problem
- Provide tools for the user
- Test for significance of the rules and rule sets

Theoretical analyses

- Fairly good algorithm
- Is a better one possible?
- How good will this algorithm be on future data sets
- A lower bound (skipped at least now)
- Association rules on random data sets (skipped at least now)
- sampling

Sampling for finding association rules

- two causes for complexity
- lots of attributes
- lots of rows
- potentially exponential in the number of attributes
- linear in the number of rows
- too many rows: take a sample from them
- in detail later

Extensions

- candidate generation
- rule generation
- database pass
  - inverted structures
  - Partition method
  - hashing to determine which candidates match a row or to prune candidates
- item hierarchies
- attributes with continuous values