Chapter 6: Episode discovery process

6. Episode discovery process

- The knowledge discovery process
- KDD process of analyzing alarm sequences
- Discovery and post-processing of large pattern collections
- TASA, Telecommunication Alarm Sequence Analyzer

The knowledge discovery process

Goal: discovery of useful and interesting knowledge

1. Understanding the domain
2. Collecting and cleaning data
3. Discovery of patterns
4. Presentation and analysis of results
5. Making onclusions and utilizing results

Pattern discovery is only a part of the KDD process (but the central one)

The knowledge discovery process

Questions implied by the KDD process model:

- How to know what could be interesting?
- How to ensure that correct and reliable discoveries can be made?
- How to discover potentially interesting patterns?
- How to make the results understandable for the user?
- How to use the results?

Episode discovery process for alarm sequences

Collecting and cleaning the data

- Can take a lot of time
- Collection of alarms rather easy
- Data cleaning? Inaccuracy of clocks
- Missing data?
- What are the event types?
  - Alarm type? Network element? A combination of the two?
- How to deal with background knowledge: network topology, object hierarchies for network elements
- "Alarm predicates": properties of alarms

Discovery of patterns

Strategy:

1. Find *all* potentially interesting patterns
   $\Rightarrow$ lots of rules
2. Allow users to explore the patterns iteratively and interactively

1. All potentially interesting patterns
   - Episodes: combination of alarms
   - Association rules: what are alarms like
   - Frequency and confidence thresholds
   - Background knowledge coded into alarm predicates in various alternative ways
   - Network topology used to constrain patterns

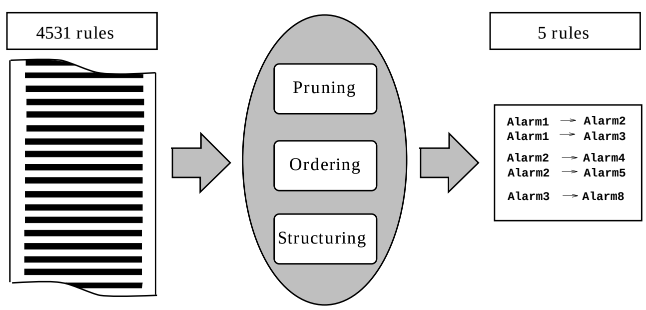
Presentation and analysis of results

There can be lots of rules

- only a small part is really interesting
- – subjective
  – hard to define in advance
  – can depend on the case
- also expected regularities (or their absence) can be of interest

⇒ iteration is necessary

⇒ support for personal views is needed



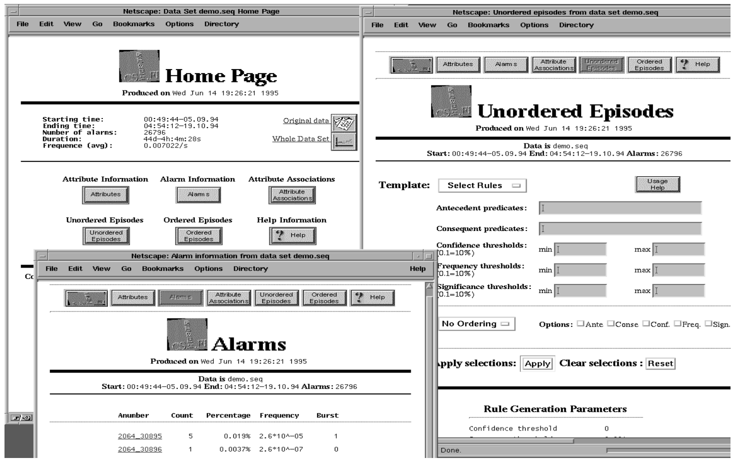
Pruning and ordering:

- alarm predicates on the left or right side
- confidence, frequency, statistical significance

Structuring:

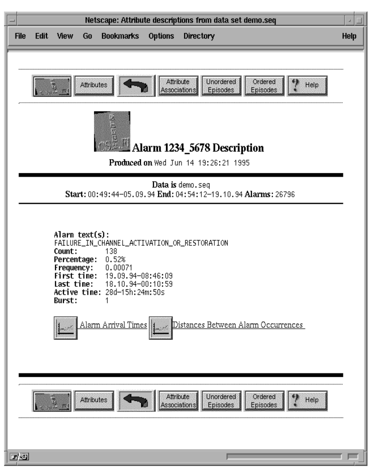
- clusters, hierarchies, etc.

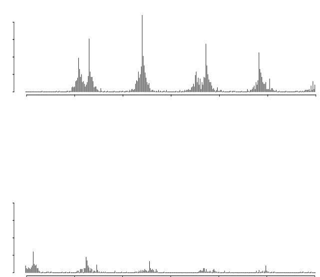
TASA: A KDD tool for alarm analysis



- pages are created automatically from analysis results

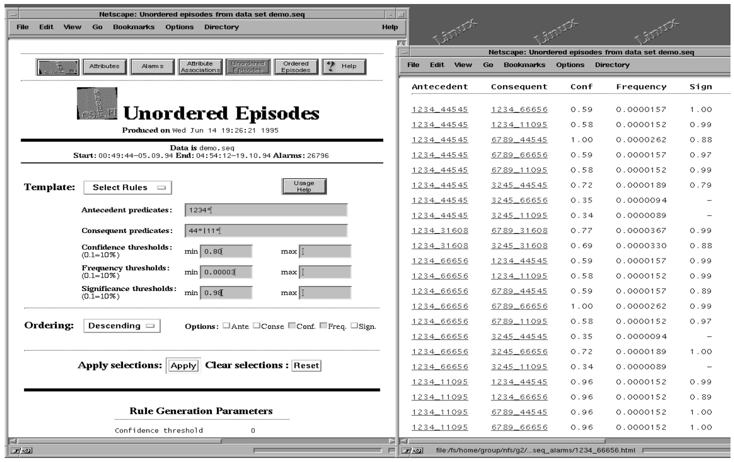
TASA: Giving an overview of data





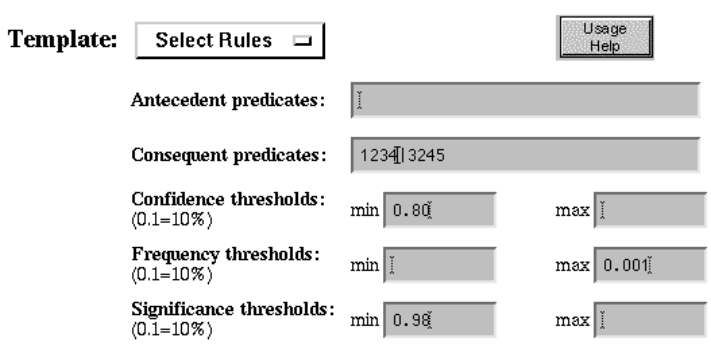
statistical information, histograms

TASA: Rule presentation



episode and association rules, views, histograms

TASA: Views with templates



- select/prune rules by their contents
  ⇒ **iteration!**
- criteria: left-hand/right-hand side of the rule, thresholds

Chapter 7: Generalized framework

7. Generalized framework

- given a set of patterns, a selection criterion, and a database
- find those patterns that satisfy the criterion in the database
- what has to be required from the patterns
- a general levelwise algorithm
- analysis in Chapter 8

Relational databases

- a *relation schema* R is a set $\{A_1, \ldots, A_m\}$ of *attributes*.
- each attribute A_i has a *domain* $Dom(A_i)$
- a *row* over a R is a sequence $\langle a_1, \ldots, a_m \rangle$ such that $a_1 \in Dom(A_i)$ for all $i = 1, \ldots, m$
- the ith value of t is denoted by $t[A_i]$
- a *relation* over R is a set of rows over R
- a *relational database* is a set of relations over a set of relation schema (the *database schema*)

Discovery task

- $\mathcal{P}$ is a set of *patterns*
- q is a *selection criterion*, i.e., a predicate $q : \mathcal{P} \times \{\mathbf{r} \mid \mathbf{r} \text{ is a database}\} \to \{\text{true, false}\}$.
- φ is *selected* if $q(\varphi, \mathbf{r})$ is true
- *frequent* as a synonym for "selected".
- give a database $\mathbf{r}$, the *theory* $\mathcal{T}(\mathcal{P}, \mathbf{r}, q)$ of $\mathbf{r}$ with respect to $\mathcal{P}$ and q is $\mathcal{T}(\mathcal{P}, \mathbf{r}, q) = \{\varphi \in \mathcal{P} \mid q(\varphi, \mathbf{r}) \text{ is true}\}$.

Example

finding all frequent item sets

- a set R a binary database r over R, a frequency threshold *min_fr*
- $\mathcal{P} = \{X \mid X \subseteq R\}$,
- $q(\varphi, r) =$ true if and only if $\mathit{fr}(\varphi, r) \geq \mathit{min\_fr}$

Selection predicate

- no semantics given for the patterns
- selection criterion takes care of that
- "$q(\varphi, \mathbf{r})$ is true" can mean different things:
- φ occurs often enough in $\mathbf{r}$
- φ is true or almost true in $\mathbf{r}$
- φ defines, in some way, an interesting property or subgroup of $\mathbf{r}$
- determining the theory of $\mathbf{r}$ is not tractable for arbitrary sets $\mathcal{P}$ and predicates q

Methodological point

- find all patterns that are selected by a relatively simple criterion—such as exceeding a frequency threshold—in order to efficiently identify a space of potentially interesting patterns
- other criteria can then be used for further pruning and processing of the patterns
- e.g., association rules or episode rules

Specialization relation

- $\mathcal{P}$ be a set of patterns, q a selection criterion over $\mathcal{P}$
- $\preceq$ a partial order on the patterns in $\mathcal{P}$
- if for all databases $\mathbf{r}$ and patterns $\varphi, \theta \in \mathcal{P}$ we have that $q(\varphi, \mathbf{r})$ and $\theta \preceq \varphi$ imply $q(\theta, \mathbf{r})$,
- then $\preceq$ is a *specialization relation* on $\mathcal{P}$ with respect to q
- $\theta \preceq \varphi$, then φ is said to be *more special* than θ and θ to be *more general* than φ
- $\theta \prec \varphi$: $\theta \preceq \varphi$ and not $\varphi \preceq \theta$
- the set inclusion relation $\subseteq$ is a specialization relation for frequent sets

Generic levelwise algorithm

- the *level* of a pattern φ in $\mathcal{P}$, denoted *level*(φ), is 1 if there is no θ in $\mathcal{P}$ for which $\theta \prec \varphi$.
- otherwise *level*(φ) is $1 + L$, where L is the maximum level of patterns θ in $\mathcal{P}$ for which $\theta \prec \varphi$
- the collection of frequent patterns of level l is denoted by $\mathcal{T}_l(\mathcal{P}, \mathbf{r}, q) = \{\varphi \in \mathcal{T}(\mathcal{P}, \mathbf{r}, q) \mid \textit{level}(\varphi) = l\}$.

Algorithm 7.6

Input: A database schema $\mathbf{R}$, a database $\mathbf{r}$ over $\mathbf{R}$, a finite set $\mathcal{P}$ of patterns, a computable selection criterion q over $\mathcal{P}$, and a computable specialization relation $\preceq$ on $\mathcal{P}$.

Output: The set $\mathcal{T}(\mathcal{P}, \mathbf{r}, q)$ of all frequent patterns.

Method:

1. compute $\mathcal{C}_1 := \{\varphi \in \mathcal{P} \mid \textit{level}(\varphi) = 1\}$;
2. $l := 1$;
3. **while** $\mathcal{C}_l \neq \emptyset$ **do**
4. // Database pass:
5. compute $\mathcal{T}_l(\mathcal{P}, \mathbf{r}, q) := \{\varphi \in \mathcal{C}_l \mid q(\varphi, \mathbf{r})\}$;
6. $l := l + 1$;
7. // Candidate generation:
8. compute $\mathcal{C}_l := \{\varphi \in \mathcal{P} \mid \textit{level}(\varphi) = l$ and $\theta \in \mathcal{T}_{\textit{level}(\theta)}(\mathcal{P}, \mathbf{r}, q)$ for all $\theta \in \mathcal{P}$ such that $\theta \prec \varphi\}$;
9. **for** all l **do** output $\mathcal{T}_l(\mathcal{P}, \mathbf{r}, q)$;

Theorem 7.7 Algorithm 7.6 works correctly. □

Examples

- association rules
- episodes: specialization relation
- exact database rules

Chapter 8: Complexity of finding frequent patterns

8. Complexity of finding frequent patterns

- border of a theory
- time usage
- guess-and-correct algorithm
- analysis
- borders and hypergraph transversals

The border of a theory

- $\mathcal{T}(\mathcal{P}, \mathbf{r}, q)$ of $\mathcal{P}$
- the whole theory can be specified by giving only the maximally specific patterns in $\mathcal{T}(\mathcal{P}, \mathbf{r}, q)$
- collection of maximally specific patterns in $\mathcal{T}(\mathcal{P}, \mathbf{r}, q)$

Definition of border

- collection of *minimally* specific (i.e., maximally general) patterns *not* in $\mathcal{T}(\mathcal{P}, \mathbf{r}, q)$
- $\mathcal{P}$ be a set of patterns, $\mathcal{S}$ a subset of $\mathcal{P}$, $\preceq$ a partial order on $\mathcal{P}$
- $\mathcal{S}$ closed downwards under the relation $\preceq$: if $\varphi \in \mathcal{S}$ and $\gamma \preceq \varphi$, then $\gamma \in \mathcal{S}$
- *border* $\mathcal{B}d(\mathcal{S})$ of $\mathcal{S}$ consists of those patterns φ such that all more general patterns than φ are in $\mathcal{S}$ and no pattern more specific than φ is in $\mathcal{S}$:

$$\begin{aligned}\mathcal{B}d(\mathcal{S}) \quad = \quad & \{\varphi \in \mathcal{P} \mid \text{for all } \gamma \in \mathcal{P} \text{ such that } \gamma \prec \varphi \text{ we have } \gamma \in \mathcal{S}, \\ & \text{and for all } \theta \in \mathcal{P} \text{ such that } \varphi \prec \theta \text{ we have } \theta \notin \mathcal{S}\}.\end{aligned}$$

- *positive border* $\mathcal{B}d^+(\mathcal{S})$

  $\mathcal{B}d^+(\mathcal{S}) = \{\varphi \in \mathcal{S} \,\big|\, \text{for all } \theta \in \mathcal{P} \text{ such that } \varphi \prec \theta \text{ we have } \theta \notin \mathcal{S}\},$

- the *negative border* $\mathcal{B}d^-(\mathcal{S})$

  $\mathcal{B}d^-(\mathcal{S}) = \{\varphi \in \mathcal{P}\backslash\mathcal{S} | \text{for all } \gamma \in \mathcal{P} \text{ such that } \gamma \prec \varphi \text{ we have } \gamma \in \mathcal{S}\}.$

Example for frequent sets

- $R = \{A, \ldots, F\}$

  $\{\{A\}, \{B\}, \{C\}, \{F\}, \{A, B\}, \{A, C\}, \{A, F\}, \{C, F\}, \{A, C, F\}\}.$

- the negative border is thus

$$\mathcal{B}d^-(\mathcal{F}) = \{\{D\}, \{E\}, \{B, C\}, \{B, F\}\}$$

- the positive border, in turn, contains the maximal frequent sets, i.e.,

$$\mathcal{B}d^+(\mathcal{F}) = \{\{A, B\}, \{A, C, F\}\}$$

- frequent episodes in a sequence over events $A, \ldots, D$

Example for strings

- $\mathcal{P}$: substrings over an alphabet $\sum$
- q: how frequently the substring occurs
- (substrings vs. subsequences $\approx$ sequential episodes)
- $\sum = \{a, b, c\}$
- $\mathcal{F} = \{a, b, c, ab, bc, abc, cb\}$
- positive border $\{abc, cb\}$
- negative border $\{ca, aa, bb, ba, cc, ac\}$(?)

Example for episodes

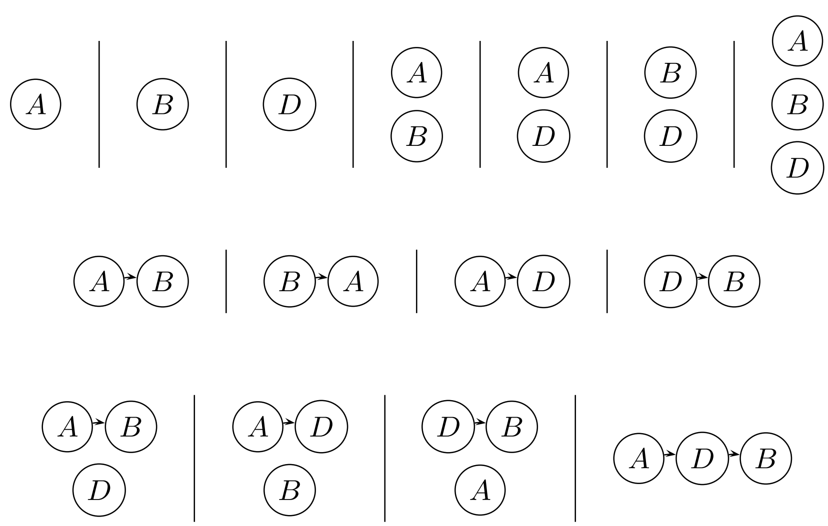


Figure 8.1: A collection $\mathcal{F}(\mathrm{s}, \textit{win}, \textit{min\_fr})$ of frequent episodes.

Figure 8.2: The positive border $\mathcal{B}d^+(\mathcal{F}(\mathbf{s}, \mathit{win}, \mathit{min\_fr}))$.

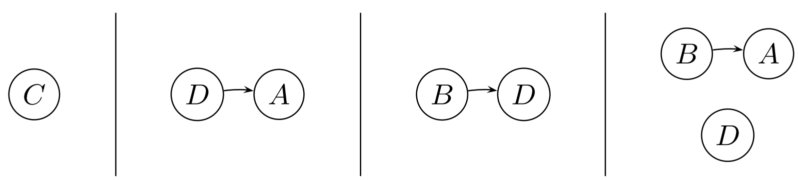


Figure 8.3: The negative border $\mathcal{B}d^-(\mathcal{F}(\mathbf{s}, \mathit{win}, \mathit{min\_fr}))$. (Tends to be tricky to check!)

Generic algorithm, again

Algorithm 7.6

Input: A database schema $\mathbf{R}$, a database $\mathbf{r}$ over $\mathbf{R}$, a finite set $\mathcal{P}$ of patterns, a computable selection criterion q over $\mathcal{P}$, and a computable specialization relation $\preceq$ on $\mathcal{P}$.

Output: The set $\mathcal{T}(\mathcal{P}, \mathbf{r}, q)$ of all frequent patterns..

Method:
1. compute $\mathcal{C}_1 := \{\varphi \in \mathcal{P} \mid \textit{level}(\varphi) = 1\}$;
2. $l := 1$;
3. **while** $\mathcal{C}_l \neq \emptyset$ **do**
4. // Database pass:
5. compute $\mathcal{T}_l(\mathcal{P}, \mathbf{r}, q) := \{\varphi \in \mathcal{C}_l \mid q(\varphi, \mathbf{r})\}$;
6. $l := l + 1$;
7. // Candidate generation:
8. compute $\mathcal{C}_l := \{\varphi \in \mathcal{P} \mid \textit{level}(\varphi) = l$ and $\theta \in \mathcal{T}_{\textit{level}(\theta)}(\mathcal{P}, \mathbf{r}, q)$ for all $\theta \in \mathcal{P}$ such that $\theta \prec \varphi\ \}$;
9. **for** all l **do** output $\mathcal{T}_l(\mathcal{P}, \mathbf{r}, q)$

Complexity of the generic algorithm

Theorem 8.4 Let $\mathcal{P}, \mathbf{r}$, and q be as in Algorithm 7.6. Algorithm 7.6 evaluates the predicate q exactly on the patterns in $\mathcal{T}(\mathcal{P}, \mathbf{r}, q) \cup \mathcal{B}d^-(\mathcal{T}(\mathcal{P}, \mathbf{r}, q))$. □

Corollary 8.5 Given a set R, a binary database r over R, and a frequency threshold *min_fr*, Algorithm 2.14 evaluates the frequency of sets in $\mathcal{F}(r, \textit{min\_fr}) \cup \mathcal{B}d^-(\mathcal{F}(r, \textit{min\_fr}))$. □

candidate generation: computes the negative border

p	*min_fr*	$\lvert\mathcal{T}(\mathcal{P},\mathbf{r},q)\rvert$	$\lvert\mathcal{B}d^{+}(\mathcal{T}(\mathcal{P},\mathbf{r},q))\rvert$	$\lvert\mathcal{B}d^{-}(\mathcal{T}(\mathcal{P},\mathbf{r},q))\rvert$
0.2	0.01	469	273	938
0.2	0.005	1291	834	3027
0.5	0.1	1335	1125	4627
0.5	0.05	5782	4432	11531

Table 8.1: Experimental results with random data sets.

min_fr	$\lvert\mathcal{T}(\mathcal{P},\mathbf{r},q)\rvert$	$\lvert\mathcal{B}d^{+}(\mathcal{T}(\mathcal{P},\mathbf{r},q))\rvert$	$\lvert\mathcal{B}d^{-}(\mathcal{T}(\mathcal{P},\mathbf{r},q))\rvert$
0.08	96	35	201
0.06	270	61	271
0.04	1028	154	426
0.02	6875	328	759

Table 8.2: Experimental results with a real data set.

The guess-and-correct algorithm

- levelwise search: safe but sometimes slow
- if there are frequent patterns that are far from the bottom of the specialization relation
- an alternative: start finding $\mathcal{T}(\mathcal{P}, \mathbf{r}, q)$ from an initial guess $\mathcal{S} \subseteq \mathcal{P}$, and then correcting the guess by looking at the database
- if the initial guess is good, few iterations are needed to correct the result

Algorithm 8.7

The *guess-and-correct algorithm* for finding all potentially interesting sentences with an initial guess $\mathcal{S}$.
Input: A database $\mathbf{r}$, a language $\mathcal{P}$ with specialization relation $\preceq$, a selection predicate q, and an initial guess $\mathcal{S} \subseteq \mathcal{P}$ for $\mathcal{T}(\mathcal{P}, \mathbf{r}, q)$. We assume $\mathcal{S}$ is closed under generalizations.

Output: The set $\mathcal{T}(\mathcal{P}, \mathbf{r}, q)$.

Algorithm 8.7

Method:

1. $\mathcal{C}^* := \emptyset$;
   // correct $\mathcal{S}$ downward:
2. $\mathcal{C} := \mathcal{B}d^+(\mathcal{S})$;
3. **while** $\mathcal{C} \neq \emptyset$ **do**
4. $\quad \mathcal{C}^* := \mathcal{C}^* \cup \mathcal{C}$;
5. $\quad \mathcal{S} := \mathcal{S} \setminus \{\varphi \in \mathcal{C} \mid q(\mathbf{r}, \varphi)$ is false$\}$;
6. $\quad \mathcal{C} := \mathcal{B}d^+(\mathcal{S}) \setminus \mathcal{C}^*$;
7. **od**;
   // now $\mathcal{S} \subseteq \mathcal{T}(\mathcal{P}, \mathbf{r}, q)$; expand $\mathcal{S}$ upwards:
8. $\mathcal{C} := \mathcal{B}d^-(\mathcal{S}) \setminus \mathcal{C}^*$;
9. **while** $\mathcal{C} \neq \emptyset$ **do**
10. $\quad \mathcal{C}^* := \mathcal{C}^* \cup \mathcal{C}$;
11. $\quad \mathcal{S} := \mathcal{S} \cup \{\varphi \in \mathcal{C} \mid q(\mathbf{r}, \varphi)$ is true$\}$;
12. $\quad \mathcal{C} := \mathcal{B}d^-(\mathcal{S}) \setminus \mathcal{C}^*$;
13. **od**;
14. output $\mathcal{S}$;

Lemma 8.8 Algorithm 8.7 works correctly. □

Theorem 8.9 Algorithm 8.7 uses at most

$$|(\mathcal{S} \triangle \mathcal{T}) \cup \mathcal{B}d(\mathcal{T}) \cup \mathcal{B}d^+(\mathcal{S} \cap \mathcal{T})|$$

evaluations of q, where $\mathcal{T} = \mathcal{T}(\mathcal{P}, \mathbf{r}, q)$. □

Initial guesses?

- sampling
- Take a small sample $\mathbf{s}$ from $\mathbf{r}$
- compute $\mathcal{T}(\mathcal{P}, \mathbf{r}, q)$ and use it as $\mathcal{S}$
- Applied to association rules this method produces extremely good results
- with a high probability one can discover the association rules holding in a database using only a single pass through the database
- other method: partitioning the database

Complexity analysis

Verification problem: assume somebody gives a set $\mathcal{S} \subseteq \mathcal{P}$ and claims that $\mathcal{S} = \mathcal{T}(\mathcal{P}, \mathbf{r}, q)$. How many evaluations of q are necessary for verifying this claim?

Theorem 8.10 Let $\mathcal{P}$ and $\mathcal{S} \subseteq \mathcal{P}$ be sets of patterns, $\mathbf{r}$ a database, q a selection criterion, and $\preceq$ a specialization relation. If the database $\mathbf{r}$ is accessed only using the predicate q, then determining whether $\mathcal{S} = \mathcal{T}(\mathcal{P}, \mathbf{r}, q)$ (1) requires in the worst case at least $|\mathcal{B}d(\mathcal{S})|$ evaluations of q, and (2) can be done in exactly $|\mathcal{B}d(\mathcal{S})|$ evaluations of q. $\square$

Corollary 8.11 Let $\mathcal{P}$ be a set of patterns, $\mathbf{r}$ a database, q a selection criterion, and $\preceq$ a specialization relation. Any algorithm that computes $\mathcal{T}(\mathcal{P}, \mathbf{r}, q)$ and accesses the data only with the predicate q must evaluate q on the patterns in $\mathcal{B}d(\mathcal{T}(\mathcal{P}, \mathbf{r}, q))$. $\square$

$R = \{A, \ldots, F\}$

claim: frequent sets are

$$\mathcal{S} = \{\{A\}, \{B\}, \{C\}, \{F\}, \{A, B\}, \{A, C\}, \{A, F\}, \{C, F\}, \{A, C, F\}\}.$$

verify this:

$\mathcal{B}d^+(\mathcal{S}) = \{\{A, B\}, \{A, C, F\}\}$ and
$\mathcal{B}d^-(\mathcal{S}) = \{\{D\}, \{E\}, \{B, C\}, \{B, F\}\}$

Computing the border

- $\mathcal{S}$, we can compute $\mathcal{B}d^+(\mathcal{S})$ without looking at the data $\mathbf{r}$
- the negative border $\mathcal{B}d^-(\mathcal{S})$ is also defined by $\mathcal{S}$
- finding the most general patterns in $\mathcal{P} \setminus \mathcal{S}$ can be difficult
- minimal transversals of hypergraphs can be used to determine the negative border
- R be a set; a collection $\mathcal{H}$ of subsets of R is a *simple hypergraph* on R, if no element of $\mathcal{H}$ is empty and if $X, Y \in \mathcal{H}$ and $X \subseteq Y$ imply $X = Y$
- elements of $\mathcal{H}$ are called the *edges*
- elements of R are the *vertices*

- a simple hypergraph $\mathcal{H}$ on a set R, a *transversal* T of $\mathcal{H}$ is a subset of R intersecting all the edges of $\mathcal{H}$
- T is a transversal if and only if $T \cap X \neq \emptyset$ for all $X \in \mathcal{H}$
- *minimal transversal* of $\mathcal{H}$ is a transversal T such that no $T' \subset T$ is a transversal
- $\mathcal{T}r(\mathcal{H})$

Frequent sets

- vertices R; define a simple hypergraph $\mathcal{H}$ to have as edges the *complements* of the sets in the positive border.
- for each set X in the positive border we have the set $R \setminus X$ as an edge in $\mathcal{H}$;
- $Y \subseteq R$; if there is an edge $R \setminus X$ such that $Y \cap (R \setminus X) = \emptyset$, then $Y \subseteq X$, and Y is frequent.
- if there is no such edge that the intersection is empty, then Y cannot be frequent.
- That is, Y is not frequent if and only if Y is a transversal of $\mathcal{H}$.
- Minimal transversals are now the minimal non-frequent sets, i.e., the negative border.

Thus:

Negative border = minimal transversals of the complements of the sets in the positive border

How to use this?

- what if only the maximal frequent sets are needed, but they are large?
- the levelwise algorithm does not work well
- Dualize-and-advance algorithm:
  - compute some maximal frequent sets using a randomized algorithm
  - compute minimal nonfrequent sets
  - verify them against the database
  - continue until no new sets are found

Chapter 9: Sampling

9. Sampling in knowledge discovery

- why sampling?
- what types of knowledge can be discovered using sampling?
- basic techniques of sampling (from files)
- sampling in finding association rules

Why sampling?

- lots of data
- many algorithms are worse that linear
- hunting for relatively common phenomena
- solution: take a sample from the data, and analyze it
- if necessary, confirm the findings by looking at the whole data set

What types of knowledge?

- estimating the sizes of certain subgroups
- opinion polls: about 1000 persons gives an accuracy of around 2 % points
- (the size of the population does not have an influence)
- what about very rare phenomena?
- "there exists a subgroup of 100 objects having these and these properties"
- very difficult to verify using sampling, if the population is large

Basic techniques of sampling

- sampling from a file
- given a file of N records $t_1, \ldots, tn_n$, we wish to choose K from them
- with replacement or without replacement
- with replacement:
  - for $i = 1$ to K do:
    - generate a random integer b between 1 and N
    - output record t_b
  - or sort the generated random integers into order and read the data once

Sampling without replacement, basic method

- keep a bit vector of N bits
- generate random integers b between 1 and N and mark bit b, if it is not already marked
- until K bits have been marked
- read through the bit vector and the data file, and output the selected records

Sampling without replacement, sequential method

$T := K$;
$M := N$;
$i := 1$;
while $T > 0$ **do**
  let b be a random number from $[0, 1]$;
  if $b < T/M$ **then**
    output record t_i;
    $T := T - 1$;
    $M := M - 1$;
  else
    $M := M - 1$;
  end;
end;

Correctness

- by induction on N; for $N = 0$ and $N = 1$, the correctness is clear
- assume the algorithm works for $N = N'$; we show that it works for $N = N' + 1$
- the first element of the file will be selected with probability K/N, as required
- what about the next elements? two cases: the first element was selected or it wasn't
- probability that an element will be selected is

$$\frac{K}{N}\frac{K-1}{N-1} + \frac{N-K}{N}\frac{K}{N-1} = \frac{K}{N}$$

Sampling for association rules

- Current algorithms require several database passes
- For very large databases, the I/O overhead is significant
- Random sample can give accurate results in sublinear time
- Random samples can be used to boost the discovery of exact association rules (a variant of guess-and-correct algorithm)
- Result: 1 database pass, in the worst case 2 passes

Simple random sample

Use a random sample only

- Frequent sets can be found in main memory
  ⇒ very efficient!
- Good news: approximations for frequencies and confidences are good
- Bad news: applications may require exact rules

Algorithm: first pass

Goal: Exact rules in (almost) one pass

1. Pick a random sample s from r
2. Select a lowered threshold $\mathit{low\_fr} < \mathit{min\_fr}$
3. Compute $\mathcal{S} = \mathcal{F}(s, \mathit{low\_fr})$ in main memory
   Goal: $\mathcal{S} \supseteq \mathcal{F}(r, \mathit{min\_fr})$
4. Compute the exact frequencies of sets in $\mathcal{S}$ using the rest of the database

A quick analysis

- I/O cost: sampling + 1 sequential database pass
- The method may fail (a frequent set is not in $\mathcal{S}$)
- Larger sample size $\Rightarrow$ lower failure probability
- Smaller $\mathit{low\_fr} \Rightarrow$ lower failure probability
- Smaller $\mathit{low\_fr} \Rightarrow \mathcal{S}$ is larger i.e., more sets are checked

How to deal with potential failures?

How much must the threshold be lowered?
How many sets have to be checked?

Negative border

- Recall: the border (both positive and negative) has to be evaluated to verify the result
- Assume $\mathcal{S} = \mathcal{F}(s, \mathit{low\_fr})$ has been computed from a sample s
- If any set not in $\mathcal{S}$ is actually frequent in r, then a set in $\mathcal{B}d^-(\mathcal{S})$ must be frequent

Negative border

- After sampling and computing $\mathcal{S}$, verify both $\mathcal{S}$ and $\mathcal{B}d^-(\mathcal{S})$ in the rest of the database (and obtain the exact frequencies)
- If no set in $\mathcal{B}d^-(\mathcal{S})$ is frequent, then $\mathcal{S}$ is guaranteed to contain all frequent sets
- If a set X in $\mathcal{B}d^-(\mathcal{S})$ is frequent, then a frequent superset of X might be missed

$\Rightarrow$ Second pass over the database can be necessary, if there are frequent sets in $\mathcal{B}d^-(\mathcal{S})$

Second pass

- Add the frequent sets in $\mathcal{B}d^-(\mathcal{S})$ to $\mathcal{S}$
- Repeat:
  - Recompute the negative border of $\mathcal{S}$
  - Add the new sets in the negative border to $\mathcal{S}$
- Compute the frequencies of sets in $\mathcal{S}$ in one pass over the database

Sampling as an instance of guess-and-correct

- Use a random sample to obtain a guess $\mathcal{S}$
  - Goal: $\mathcal{S} \supset \mathcal{F}(r, \mathit{min\_fr})$
  - 1st pass: correction in one direction only (removal of infrequent sets)
- Negative border $\mathcal{B}d^-(\mathcal{S})$ tells whether frequent sets were missed
  - If necessary, add all possibly frequent sets to $\mathcal{S}$
  - Now $\mathcal{S} \supset \mathcal{F}(r, \mathit{min\_fr})$ is guaranteed
  - 2nd pass: evaluate $\mathcal{S}$

Dynamic threshold

- Second pass over the database is necessary, if there are frequent sets in $\mathcal{B}d^-(\mathcal{S})$
- $\Rightarrow$ Frequencies of border sets can be used to estimate the probability of a second pass
- Idea: set the lowered threshold in run time, so that the probability of a second pass is within a desired range

Chernoff bounds

Theorem 9.8 Given an item set X and a random sample s of size

$$|s| \geq \frac{1}{2\varepsilon^2} \ln \frac{2}{\delta}$$

the probability that $|\mathit{fr}(X,s) - \mathit{fr}(X)| > \varepsilon$ is at most δ.

Proof The Chernoff bounds give the result $Pr[|x - np| > a] < 2e^{-2a^2/n}$, where x is a random variable with binomial distribution $B(n,p)$. For the probability at hand we thus have

$$\begin{aligned}
& Pr[|\mathit{fr}(X,s) - \mathit{fr}(X)| > \varepsilon] \\
= \quad & Pr[|\mathit{fr}(X,s) - \mathit{fr}(X)| \cdot |s| > \varepsilon\,|s|] \\
\leq \quad & 2e^{-2(\varepsilon\,|s|)^2/|s|} \leq \delta.
\end{aligned}$$

What does this mean?

Sufficient sample sizes (note: Chernoff bounds are rough!)

ε	δ	Sample size
0.01	0.01	27 000
0.01	0.001	38 000
0.01	0.0001	50 000
0.001	0.01	2 700 000
0.001	0.001	3 800 000
0.001	0.0001	5 000 000

Table 9.1 Sufficient sample sizes, given ε and δ.

What about several sets?

Corollary 9.9 Given a collection $\mathcal{S}$ of sets and a random sample s of size

$$|s| \geq \frac{1}{2\varepsilon^2} \ln \frac{2|\mathcal{S}|}{\Delta},$$

the probability that there is a set $X \in \mathcal{S}$ such that $|fr(X,s) - fr(X)| > \varepsilon$ is at most Δ.

Proof By Theorem 9.8, the probability that $|fr(X,s) - fr(X)| > \varepsilon$ for a given set X is at most $\frac{\Delta}{|\mathcal{S}|}$. Since there are $|\mathcal{S}|$ such sets, the probability in question is at most Δ.

Experiments

- Three benchmark data sets from [AS94]
- Assumption: real data sets can be much larger
- Sampling with replacement (analysis is easier)
- Sample sizes from 20,000 to 80,000
- Every experiment was repeated 100 times
- *low_fr* was set so that the probability of missing any given frequent set is at most 0.001

Results

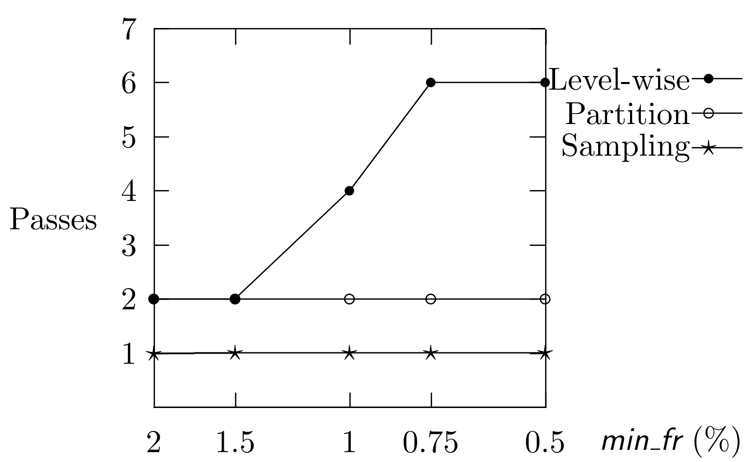


Figure 9.1 The number of database passes for frequent set algorithms (T10.I4.D100K)

Results

Lowered frequency threshold

	Sample size $|s|$			
min_fr (%)	20,000	40,000	60,000	80,000
0.25	0.13	0.17	0.18	0.19
0.50	0.34	0.38	0.40	0.41
0.75	0.55	0.61	0.63	0.65
1.00	0.77	0.83	0.86	0.88
1.50	1.22	1.30	1.33	1.35
2.00	1.67	1.77	1.81	1.84

Table 9.3 Lowered frequency thresholds for $\delta = 0.001$

Results

Number of sets checked: insignificant increase

	Sample size				
min_fr	20,000	40,000	60,000	80,000	Level-wise
0.50	382,282	368,057	359,473	356,527	318,588
0.75	290,311	259,015	248,594	237,595	188,024
1.00	181,031	158,189	146,228	139,006	97,613
1.50	52,369	40,512	36,679	34,200	20,701
2.00	10,903	7,098	5,904	5,135	3,211

Table 9.5 Number of itemsets considered for data set T10.I4.D100K

Exact I/O savings?

- Depends on storage structures and sampling methods
- Example 1:
  Database size 10 million rows,
  sample size 20 thousand rows,
  100 rows/disk block
  $\Rightarrow$ sampling reads at most 20 % of the database
- Example 2:
  database size 10 billion rows
  $\Rightarrow$ sampling reads at most 0.02 % of the database