Chapter 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 conclusions and utilizing results

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

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?
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
   ⇒ 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
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 $\text{Dom}(A_i)$
- a row over a $R$ is a sequence $(a_1, \ldots, a_m)$ such that $a_1 \in \text{Dom}(A_i)$ for all $i = 1, \ldots, m$
- the $i$th 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 $q : \mathcal{P} \times \{r \mid r \text{ is a database}\} \rightarrow \{\text{true, false}\}$.
- $\varphi$ is selected if $q(\varphi, r)$ is true
- frequent as a synonym for "selected".
- give a database $r$, the theory $T(\mathcal{P}, r, q)$ of $r$ with respect to $\mathcal{P}$ and $q$ is $T(\mathcal{P}, r, q) = \{\varphi \in \mathcal{P} \mid q(\varphi, r) \text{ is true}\}$. 
Example

finding all frequent item sets

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

Selection predicate

- no semantics given for the patterns
- selection criterion takes care of that
- “$q(\varphi, r)$ is true” can mean different things:
  - $\varphi$ occurs often enough in $r$
  - $\varphi$ is true or almost true in $r$
  - $\varphi$ defines, in some way, an interesting property or subgroup of $r$
  - determining the theory of $r$ 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 \( r \) and patterns \( \varphi, \theta \in \mathcal{P} \) we have that \( q(\varphi, r) \) and \( \theta \preceq \varphi \) imply \( q(\theta, r) \),
- then \( \preceq \) is a specialization relation on \( \mathcal{P} \) with respect to \( q \)
- \( \theta \preceq \varphi \), then \( \varphi \) is said to be more special than \( \theta \) and \( \theta \) to be more general than \( \varphi \)
- \( \theta \prec \varphi : \theta \preceq \varphi \) 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 $\varphi$ in $\mathcal{P}$, denoted $\text{level}(\varphi)$, is 1 if there is no $\theta$ in $\mathcal{P}$ for which $\theta \prec \varphi$.
- otherwise $\text{level}(\varphi)$ is $1 + L$, where $L$ is the maximum level of patterns $\theta$ in $\mathcal{P}$ for which $\theta \prec \varphi$.
- the collection of frequent patterns of level $l$ is denoted by $T_l(\mathcal{P}, r, q) = \{ \varphi \in T(\mathcal{P}, r, q) \mid \text{level}(\varphi) = l \}$.

Algorithm 7.6

**Input:** A database schema $R$, a database $r$ over $R$, 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 $T(\mathcal{P}, r, q)$ of all frequent patterns.

**Method:**
1. compute $C_1 := \{ \varphi \in \mathcal{P} \mid \text{level}(\varphi) = 1 \}$;
2. $l := 1$;
3. while $C_l \neq \emptyset$ do
4.   // Database pass:
5.   compute $T_l(\mathcal{P}, r, q) := \{ \varphi \in C_l \mid q(\varphi, r) \}$;
6.   $l := l + 1$;
7.   // Candidate generation:
8.   compute $C_l := \{ \varphi \in \mathcal{P} \mid \text{level}(\varphi) = l \text{ and } \exists \theta \in T_{\text{level}(\varphi)}(\mathcal{P}, r, q) \text{ for all } \theta \in \mathcal{P} \text{ such that } \theta \prec \varphi \}$;
9.   for all $l$ do output $T_l(\mathcal{P}, r, q)$;
Theorem 7.7 Algorithm 7.6 works correctly.

Examples

- association rules
- episodes: specialization relation
- exact database rules
Chapter 8: Complexity of finding frequent patterns

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

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

**Definition of border**

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

$$Bd(S) = \{ \varphi \in P \mid \text{for all } \gamma \in P \text{ such that } \gamma \prec \varphi \text{ we have } \gamma \in S, \text{ and for all } \theta \in P \text{ such that } \varphi \prec \theta \text{ we have } \theta \notin S \}.$$
• positive border $Bd^+(S)$
  
  \[ Bd^+(S) = \{ \varphi \in S \mid \text{for all } \theta \in P \text{ such that } \varphi < \theta \text{ we have } \theta \notin S \} \]

• the negative border $Bd^-(S)$
  
  \[ Bd^-(S) = \{ \varphi \in P \setminus S \mid \text{for all } \gamma \in P \text{ such that } \gamma < \varphi \text{ we have } \gamma \in S \} \]

Algorithmic Methods of Data Mining, Fall 2005, Chapter 8: Complexity of finding frequent patterns

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
  
  \[ Bd^-(\mathcal{F}) = \{ \{D\}, \{E\}, \{B, C\}, \{B, F\} \} \]

• the positive border, in turn, contains the maximal frequent sets, i.e.,
  
  \[ Bd^+(\mathcal{F}) = \{ \{A, B\}, \{A, C, F\} \} \]

• 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 $\{ca, aa, bb, ba, cc, ac\}$?

Algorithmic Methods of Data Mining, Fall 2005, Chapter 8: Complexity of finding frequent patterns

Example for episodes

Figure 8.1: A collection $\mathcal{F}(s, win, min, fr)$ of frequent episodes.
Figure 8.2: The positive border $Bd^+ (F(s, \text{win}, \text{min}_fr))$.

Figure 8.3: The negative border $Bd^- (F(s, \text{win}, \text{min}_fr))$. (Tends to be tricky to check!)
Algorithm 7.6

Input: A database schema $R$, a database $r$ over $R$, 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 $T(\mathcal{P}, r, q)$ of all frequent patterns.

Method:
1. compute $C_1 := \{ \varphi \in \mathcal{P} \mid \text{level}(\varphi) = 1 \}$ ;
2. $l := 1$;
3. while $C_l \neq \emptyset$ do
   4. // Database pass:
      5. compute $T_l(\mathcal{P}, r, q) := \{ \varphi \in C_l \mid q(\varphi, r) \}$ ;
      6. $l := l + 1$;
   7. // Candidate generation:
      8. compute $C_l := \{ \varphi \in \mathcal{P} \mid \text{level}(\varphi) = l \text{ and } \theta \in T_{\text{level}(\theta)}(\mathcal{P}, r, q) \text{ for all } \theta \in \mathcal{P} \text{ such that } \theta \prec \varphi \}$;
9. for all $l$ do output $T_l(\mathcal{P}, r, q)$
Theorem 8.4 Let $P$, $r$, and $q$ be as in Algorithm 7.6. Algorithm 7.6 evaluates the predicate $q$ exactly on the patterns in $T(P, r, q) \cup Bd^{-}(T(P, r, q))$.

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 $F(r, min_{fr}) \cup Bd^{-}(F(r, min_{fr}))$.

candidate generation: computes the negative border
| $p$   | $\text{min}_fr$ | $|T(P, r, q)|$ | $|Bd^+(T(P, r, q))|$ | $|Bd^-(T(P, r, q))|$ |
|-------|-----------------|----------------|----------------|----------------|
| 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.

| $\text{min}_fr$ | $|T(P, r, q)|$ | $|Bd^+(T(P, r, q))|$ | $|Bd^-(T(P, r, q))|$ |
|-----------------|----------------|----------------|----------------|
| 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 $T(\mathcal{P}, r, q)$ from an initial guess $S \subseteq \mathcal{P}$, 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 $S$.

**Input:** A database $r$, a language $\mathcal{P}$ with specialization relation $\preceq$, a selection predicate $q$, and an initial guess $S \subseteq \mathcal{P}$ for $T(\mathcal{P}, r, q)$. We assume $S$ is closed under generalizations.

**Output:** The set $T(\mathcal{P}, r, q)$. 
Algorithm 8.7
Method:
1. \( C^* := \emptyset; \)
   // correct \( S \) downward:
2. \( C := Bd^+(S); \)
3. while \( C \neq \emptyset \) do
4. \( C^* := C^* \cup C; \)
5. \( S := S \setminus \{\varphi \in C \mid q(r, \varphi) \text{ is false}\}; \)
6. \( C := Bd^+(S) \setminus C^*; \)
7. od;
   // now \( S \subseteq T(P, r, q) \); expand \( S \) upwards:
8. \( C := Bd^-(S) \setminus C^*; \)
9. while \( C \neq \emptyset \) do
10. \( C^* := C^* \cup C; \)
11. \( S := S \cup \{\varphi \in C \mid q(r, \varphi) \text{ is true}\}; \)
12. \( C := Bd^-(S) \setminus C^*; \)
13. od;
14. output \( S; \)

Lemma 8.8 Algorithm 8.7 works correctly. □

Theorem 8.9 Algorithm 8.7 uses at most

\[ |(S \Delta T) \cup Bd(T) \cup Bd^+(S \cap T)| \]

evaluations of \( q \), where \( T = T(P, r, q). \) □
Initial guesses?

- sampling
- Take a small sample $s$ from $r$
- compute $T(\mathcal{P}, r, q)$ and use it as $S$
- 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

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

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

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

\[ R = \{A, \ldots, F\} \]

claim: frequent sets are

\[ S = \{\{A\}, \{B\}, \{C\}, \{F\}, \{A, B\}, \{A, C\}, \{A, F\}, \{C, F\}, \{A, C, F\}\}. \]

verify this:

\[ Bd^+(S) = \{\{A, B\}, \{A, C, F\}\} \text{ and} \]
\[ Bd^-(S) = \{\{D\}, \{E\}, \{B, C\}, \{B, F\}\}. \]
• a simple hypergraph $H$ on a set $R$, a transversal $T$ of $H$ is a subset of $R$ intersecting all the edges of $H$

$T$ is a transversal if and only if $T \cap X \neq \emptyset$ for all $X \in H$

• minimal transversal of $H$ is a transversal $T$ such that no $T' \subset T$ is a transversal

$Tr(H)$

Frequent sets

• vertices $R$; define a simple hypergraph $H$ 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 $H$;

• $Y \subseteq R$; 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 $H$.

• 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 than 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 $t_1, \ldots, t_n$, 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 \( low_{fr} < min_{fr} \)
3. Compute \( S = F(s, low_{fr}) \) in main memory
   Goal: \( S \supseteq F(r, min_{fr}) \)
4. Compute the exact frequencies of sets in \( S \) 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 \( S \))
- Larger sample size \( \Rightarrow \) lower failure probability
- Smaller \( low_{fr} \) \( \Rightarrow \) lower failure probability
- Smaller \( low_{fr} \) \( \Rightarrow \) \( S \) 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 $S = \mathcal{F}(s, \text{low}_fr)$ has been computed from a sample $s$
- If any set not in $S$ is actually frequent in $r$, then a set in $Bd^-(S)$ must be frequent

- After sampling and computing $S$, verify both $S$ and $Bd^-(S)$ in the rest of the database (and obtain the exact frequencies)
- If no set in $Bd^-(S)$ is frequent, then $S$ is guaranteed to contain all frequent sets
- If a set $X$ in $Bd^-(S)$ 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 $Bd^-(S)$
Second pass

- Add the frequent sets in $Bd^-(S)$ to $S$
- Repeat:
  - Recompute the negative border of $S$
  - Add the new sets in the negative border to $S$
- Compute the frequencies of sets in $S$ in one pass over the database

Sampling as an instance of guess-and-correct

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

- Second pass over the database is necessary, if there are frequent sets in $Bd^-(S)$
- $\Rightarrow$ 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 $|fr(X, s) - fr(X)| > \varepsilon$ is at most $\delta$.

**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

$$Pr[|fr(X, s) - fr(X)| > \varepsilon]$$

$$= Pr[|fr(X, s) - fr(X)| \cdot |s| > \varepsilon |s|]$$

$$\leq 2e^{-2(\varepsilon |s|)^2/|s|} \leq \delta.$$
Sufficient sample sizes (note: Chernoff bounds are rough!)

<table>
<thead>
<tr>
<th>$\varepsilon$</th>
<th>$\delta$</th>
<th>Sample size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>0.01</td>
<td>27 000</td>
</tr>
<tr>
<td>0.01</td>
<td>0.001</td>
<td>38 000</td>
</tr>
<tr>
<td>0.01</td>
<td>0.0001</td>
<td>50 000</td>
</tr>
<tr>
<td>0.001</td>
<td>0.01</td>
<td>2 700 000</td>
</tr>
<tr>
<td>0.001</td>
<td>0.001</td>
<td>3 800 000</td>
</tr>
<tr>
<td>0.001</td>
<td>0.0001</td>
<td>5 000 000</td>
</tr>
</tbody>
</table>

Table 9.1 Sufficient sample sizes, given $\varepsilon$ and $\delta$.

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 $\Delta$.

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 $\Delta$. 
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
- $\text{low}_{fr}$ was set so that the probability of missing any given frequent set is at most 0.001

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

| $min_{fr}$ (%) | Sample size $|s|$ | 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 $\delta = 0.001$

## Number of sets checked: insignificant increase

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

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