Course overview

581550-4 Data Mining (3 cu)

- Data mining, also called knowledge discovery in databases (KDD)
- In Finnish: tiedon louhinta, tietämyksen muodostaminen
- Goal of the course: an overview of pattern discovery
- Biased overview
- Theory and examples
- Course home page:
  http://www.cs.helsinki.fi/hannu.toivonen/teaching/timuS02/

Contents of the course

- Introduction
- Discovery of association rules
- Case study: discovery of episodes from telecommunications alarm databases
- Theoretical aspects of knowledge discovery
- Closed sets and formal concept analysis
- Sampling
- Search for integrity constraints in databases

Course organization

- lectures
  - Tuesdays and Thursdays 10–12 A414
  - 10 Sep – 17 Oct
  - language of instruction: Finnish(?)
  - (discussions: Finnish or English)
- exercises
  - Tue 8–10 A320 (in English?), Fri 14–16 B450
  - 17 Sep – 24 Oct
  - (a new group needed?)

Course organization (cont)

- project work
  - programming, data analysis or essays
  - acounts for appr. 1/3 of the course (1 cu)
  - detailed instructions will be given later
  - due by Fri 25 Oct

Course organization (cont)

- course exam: Fri 1 Nov 02, 14:00–18:00, Auditorium
- project works due: Fri 25 Oct 02
- course assistant: Taneli Mielikäinen

- project work: 0–20 points, compulsory, at least 10 points required to pass the course
- exam: 0–40 points, compulsory, at least 20 points required to pass the course
- weekly exercises: 0–10 extra points
- minimum 30/60 points, maximum 70/60 points!

Material

- course notes: Heikki Mannila and Hannu Toivonen: Knowledge Discovery in Databases: Search for Frequent Patterns
- original articles
- copies of slides
- available on the web during the course
- hardcopies available for copying in room A412

Background check

- prerequisities: cum laude, some mathematics
- design and analysis of algorithms (algoritmien suunnittelu ja analyysi)
- machine learning (koneoppiminen)
- approbatur or more in statistics
- probability I (todennäköisyyslaskenta I)
- database structures and algorithms (tietokannanhallinta tai tiha II)
- working on or finished a M.Sc. thesis
- looking for a M.Sc. thesis topic

Data mining activities at UH

- basic research: algorithms, theory
- applied research: genetics, ecology, ubiquitous computing, documents, natural language, ...
- FDK "From Data to Knowledge": center of excellence 2002–2007
- HIIT Basic Research Unit

Chapter 1: Introduction

Chapter 1. Introduction

- Introduction
- What is KDD?
- Two examples
- Data mining vs. statistics and machine learning
- Data mining and databases

What is data mining?

Goal: obtain useful knowledge
from large masses of data.

- "Tell something interesting about this data."
- "Describe this data."
- exploratory data analysis on large data sets

Examples of discovered knowledge

- association rules:
  "80 % of customers who buy beer and sausage buy also mustard"
- rules: "if Age $<$ 40 then Income $<$ 10"
- functional dependencies:
  $A \rightarrow B$, i.e., "if $t[A] = u[A]$, then $t[B] = u[B]$"
- belief networks
- clusterings

Example: sales data

- 0/1 matrix D
- rows:customers
- columns: products
- $t(A) = 1$ iff customer t bought product A
- thousands of products, millions of rows
- easy to collect
- find something interesting from this?

Association rules

- mustard, sausage, beer $\Rightarrow$ chips
- conditional probability (*confidence*) of the rule: 0.8
- *frequency* of the rule: 0.2
- arbitrary number of conjuncts on the left-hand side

Find all association rules with frequency at least *min_fr*.

Example: student/course data

- matrix D with $D(t, c) = 1$ iff student t has taken course c

```
1 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 1 0 0 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
1 1 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Example: student/course data

- example rule:
  `if {Data Communications, Unix, Networks}`
  `then (graduation) (0.3)`

Example: SKICAT sky survey

approximately 1000 photographs of 13000x13000 pixels

$\Rightarrow 3 \cdot 10^9$ smudges of light, each with 40 attributes

- task 1: classify the objects to
  - stars
  - galaxies
  - others
- task 2: find some interesting clusters of objects
  - quasars with high redshift
  - ...

machine learning? pattern recognition?

Why data mining?

- raw data is easy to collect, expensive to analyze
- more data than can be analyzed using traditional methods
- suspicion that important knowledge could be there
- successful applications
- methods: machine learning, statistics, databases
- a tool for exploratory data analysis
- can and has to be used in combination with traditional methods
- strong interest from 1989–, hype from 1995–

The KDD process

- understanding the domain,
- preparing the data set,
- discovering patterns (data mining),
- postprocessing of discovered patterns, and
- putting the results into use

Where is the effort?

iteration, interaction

Data mining vs. data warehousing and OLAP

Very briefly:

- data warehousing makes data mining a lot cheaper
- data mining is one of the reasons for data warehousing
- OLAP: verification-driven
- data mining: discovery-driven
- MIS (management information systems)

Data mining and related areas

Possible comments:

- That's just machine learning!
- That's just statistics!
- What has that to do with databases?

Data mining vs. machine learning

- machine learning methods are used for data mining
  - classification, clustering, ...
- amount of data makes a difference
  - accessing examples can be a problem
- data mining has more modest goals:
  - automating tedious discovery tasks, not aiming at human performance in real discovery
  - helping user, not replacing them
- this course: data mining $\setminus$ machine learning

Data mining vs. statistics

"tell me something interesting about this data" – what else is this than statistics?

- the goal is similar
- different types of methods
- in data mining one investigates a lot of possible hypotheses
- amount of data
- data mining as a preliminary stage for statistical analysis
- challenge to data mining: better contact with statistics

Data mining and databases

- ordinary database usage: deductive
- knowledge discovery: inductive
- new challenges for database vendors
- parts of databases techniques are not very relevant
  - transaction management
  - recovery

Database research for data mining I

- making it possible to run data mining algorithms on very large databases
- modified learning algorithms
  - minimize number of passes through the data
  - modify access routes
  - ...
- data servers: architectures, algorithms, data structures, memory management
- primitive operations inside a DBMS
- several data mining applications use currently very little of what is known about efficient access structures

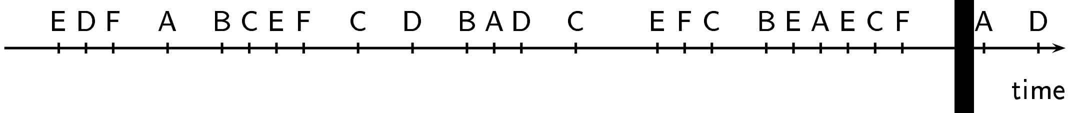
Database research for data mining II

- "Inductive databases"
- why was the relational model so successful?
  - *ad hoc* queries
- the same is needed for data mining
- high-level query languages for data mining applications
- query processing and optimization for such languages
- not easy
- Imielinski & Mannila: A database perspective on knowledge discovery, CACM November 1996.

Example: episodes in sequences

- Data: events in time, e.g.,
  - alarms in telecommunications networks
  - user actions in an user interface
  - database transactions
  - biostatistical events
- Goals: understanding the structure of the process producing the events, prediction of future events.
- How to discover which combinations of events occur frequently?

Example sequence



Observations:

- whenever E occurs, F occurs soon
- whenever A and B occur (in either order), C occurs soon

Telecommunications alarm log

Event	Time
KE82K02-31	780560888
KE82K10-16	780560892
H-M-K09-57	780560917
SOT-K01-03	780560926
MUU-K03-04	780561011
KE82K10-16	780561015
PAK-K14-27	780561119
KE82K10-16	780561138

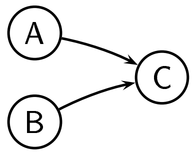
100–400 different events
events occur recurrently
1 month = 70 000 alarms

Sequences and episodes

Data a sequence of (event, time) pairs

$(A, 123), (B, 125), (D, 140), (A, 150), (C, 151), (D, 201), (A, 220)$

Patterns episodes: a set of events
occurring close to each other in time
in an order extending a given partial order



Pattern class All serial episodes / all parallel episodes / all episodes / ...

Occurrence pattern occurs frequently in data if there are sufficiently many windows of size W in the data such that the pattern occurs in the window

Discovering frequent patterns

- find all frequent episodes of size 1
- build candidate episodes of size 2
- check which episodes of size 2 occur frequently
- continue
- incremental recognition, using previous rounds of computation, ...

A general model for data mining

- given a class of patterns
- an occurrence criterion
- find all patterns from the class that occur frequently enough

Why is data mining fun?

- practically relevant
- easy theoretical questions
- the whole spectrum: from theoretical issues to systems issues to concrete data cleaning to novel discoveries
- easy to cooperate with other areas and with industry

Chapter 2: Association rules

Chapter 2. Association rules

- 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
- beer $\Rightarrow$ chips
  - confidence (conditional probability): 0.87
  - frequency (support): 0.34

Problem formulation: data

- a set R of items
- a *0/1 relation* r *over* R 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 $R = \{A, \ldots, K\}$.

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*.

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 $\{A, \ldots, K\}$

- *fr*$(\{A, B\}, r) = 3/5 = 0.6$
- $\mathcal{M}(\{A, B\}, r) = \{t_1, t_2, t_4\}$

Frequent sets

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

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

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

Association rules

- Let R be a set, r a 0/1 relation over R, and $X, Y \subseteq R$ sets of items
- $X \Rightarrow Y$ is an *association rule* over r.
- The *confidence* of $X \Rightarrow Y$ in r, denoted by $\textit{conf}(X \Rightarrow Y, r)$, is $\frac{|\mathcal{M}(X \cup Y, r)|}{|\mathcal{M}(X, r)|}$.
  - The confidence $\textit{conf}(X \Rightarrow Y, r)$ is the conditional probability that a row in r matches Y given that it matches X

Association rules II

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

Discovery task

- given R, r, *min_fr*, and *min_conf*
- find all association rules $X \Rightarrow Y$ that hold in r with respect to *min_fr* and *min_conf*
- X and Y 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, confidence 1
- when is the task feasible?

How to find association rules

- Find all frequent item sets $X \subseteq R$ and their frequencies.
- Then test separately for all $Y \subset X$ with $Y \neq \emptyset$ whether the rule $X \setminus Y \Rightarrow Y$ holds with sufficient confidence.
- 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 confidence threshold *min_conf*.

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

Method:

```
// Find frequent sets (Algorithm 50):
compute F(r, min_fr) := {X ⊆ R | fr(X, r) ≥ min_fr};
// Generate rules:
for all X ∈ F(r, min_fr) do
    for all Y ⊂ X with Y ≠ ∅ do
        if fr(X)/fr(X \ Y) ≥ min_conf then
            output the rule X \ Y ⇒ Y, fr(X), and fr(X)/fr(X \ Y);
```

Correctness and running time

- the algorithm is correct
- running time?

Finding frequent sets: reasoning behind Apriori

- trivial solution: look at all subsets of R
- 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 $Y \subseteq X$
- then $\mathcal{M}(Y) \supseteq \mathcal{M}(X)$, and $fr(Y) \geq fr(X)$
- if X is frequent then Y is also frequent
- Let $X \subseteq R$ be a set. If any of the proper subsets $Y \subset X$ is not frequent then (1) X is not frequent and (2) there is a non-frequent subset $Z \subset X$ 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: generate and test
- candidate collection:

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

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:

$\mathcal{C}_1 := \{\{A\} \mid A \in R\}$;
$l := 1$;
while $\mathcal{C}_l \neq \emptyset$ **do**
    // Database pass (Algorithm 57):
    compute $\mathcal{F}_l(r) := \{X \in \mathcal{C}_l \mid \mathit{fr}(X, r) \geq \mathit{min\_fr}\}$;
    $l := l + 1$;
    // Candidate generation (Algorithm 54):
    compute $\mathcal{C}_l := \mathcal{C}(\mathcal{F}_{l-1}(r))$;
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 $fr(\{A, B, C\}) = fr(\{A, B\})$
- can infer $fr(\{A, B, D\}) < min\_fr$
- how?

Candidate generation

- how to generate the collection $\mathcal{C}(\mathcal{F}_l(r))$?
- trivial method: check all subsets
- compute potential candidates as unions $X \cup Y$ of size $l + 1$
- here X and Y 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 $Y \in \mathcal{F}_l(r)$ such that $X < Y$ and X and Y share their $l-1$ lexicographically first items **do**
3.   **for** all $Z \subset (X \cup Y)$ such that $|Z| = l$ **do**
4.   **if** Z is not in $\mathcal{F}_l(r)$ **then** continue with the next Y at line 2;
5.   output $X \cup Y$;

Correctness and running time

Theorem 1 *Algorithm 54 works correctly.*

Theorem 2 *Algorithm 54 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 $\mathcal{C}_l \supseteq \mathcal{F}_l(r, \textit{min\_fr})$, and $\textit{min\_fr}$.

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

Method:

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

Data structures

- for each $A \in R$ a list $A.\textit{is\_contained\_in}$ of candidates that contain A
- For each candidate X we maintain two counters:
  - $X.\textit{freq\_count}$ the number of rows that X matches,
  - $X.\textit{item\_count}$ the number of items of X

Correctness

- clear (?)

Time complexity

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

Experimental results

- small course registration database
- 4 734 students
- 127 courses
- frequency thresholds 0.01–0.2

Size	Frequency threshold					
	0.200	0.100	0.075	0.050	0.025	0.010
1	6	13	14	18	22	36
2	1	21	48	77	123	240
3	0	8	47	169	375	898
4	0	1	12	140	776	2 203
5	0	0	1	64	1 096	3 805
6	0	0	0	19	967	4 899
7	0	0	0	2	524	4 774
8	0	0	0	0	165	3 465
9	0	0	0	0	31	1 845
10	0	0	0	0	1	690
11	0	0	0	0	0	164
12	0	0	0	0	0	21
13	0	0	0	0	0	1

Table 1: Number of frequent sets of each size with different frequency thresholds.

	Frequency threshold					
	0.200	0.100	0.075	0.050	0.025	0.010
Candidate sets:						
Count	142	223	332	825	4 685	24 698
Generation time (s)	0.1	0.1	0.2	0.2	1.1	10.2
Frequent sets:						
Count	7	43	122	489	4 080	23 041
Maximum size	2	4	5	7	10	13
Database pass time (s)	0.7	1.9	3.5	10.3	71.2	379.7
Match	5 %	19 %	37 %	59 %	87 %	93 %
Rules (*min_conf* $= 0.9$):						
Count	0	3	39	503	15 737	239 429
Generation time (s)	0.0	0.0	0.1	0.4	46.2	2 566.2
Rules (*min_conf* $= 0.7$):						
Count	0	40	193	2 347	65 181	913 181
Generation time (s)	0.0	0.0	0.1	0.8	77.4	5 632.8
Rules (*min_conf* $= 0.5$):						
Count	0	81	347	4 022	130 680	1 810 780
Generation time (s)	0.0	0.0	0.1	1.1	106.5	7 613.62

Different statistics of association rule discovery with course database.

	Candidates		Frequent sets		
Size	Count	Time (s)	Count	Time (s)	Match
1	127	0.05	22	0.26	17 %
2	231	0.04	123	1.79	53 %
3	458	0.04	375	5.64	82 %
4	859	0.09	776	12.92	90 %
5	1 168	0.21	1 096	18.90	94 %
6	1 058	0.30	967	18.20	91 %
7	566	0.24	524	9.69	93 %
8	184	0.11	165	3.09	90 %
9	31	0.04	31	0.55	100 %
10	3	0.01	1	0.15	33 %
11	0	0.00			
Total	4 685	1.13	4 080	71.19	87 %

Number of sets and time used for set of different sizes

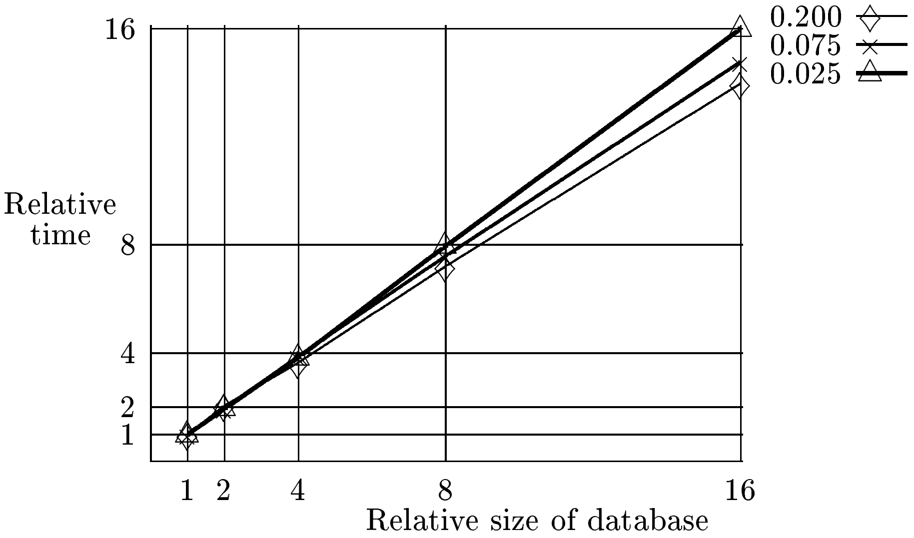


Figure 2: Results of scale-up tests.

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

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

Uninteresting rules

- There are 2010 association rules in the course enrollment database that match at least 11 students (i.e., the frequency (or support) threshold is 0.01).
- prior knowledge: Design and Analysis of Algorithms $\Rightarrow$ Introduction to Computers (0.97, 0.03).
- uninteresting attributes or attribute combinations. Introduction to Computers $\Rightarrow$ Programming in Pascal (0.95, 0.60) is useless, if the user is only interested in graduate courses.
- Rules can be redundant. Data Communications, Programming in C $\Rightarrow$ Unix Platform (0.14, 0.03) and Data Communications, Programming in C, Introduction to Unix $\Rightarrow$ Unix Platform (0.14, 0.03).

Iteration

- filter out rules referring to uninteresting courses
- all rules containing basic courses away: only half are left
- focus to, e.g., all rules containing the course “Programming in C”
- filter out “Unix Platform”
- etc.

Operations

- pruning: reduction of the number of rules;
- ordering: sorting of rules according, e.g., to statistical significance; and
- structuring: organization of the rules, e.g., to clusters or hierarchies.
- other operations?

Pruning using templates

- hierarchies among attributes {Artificial Intelligence, Programming in C, Data Communications} $\subset$ Undergraduate Course $\subset$ Any Course,
- a template is an expression $A_1, \ldots, A_k \Rightarrow A_{k+1}, \ldots, A_l$,
- A_i: an attribute name, a class name, or an expression $C+$ or $C*$
- Graduate Course, Any Course$*$ $\Rightarrow$ Design and Analysis of Algorithms
- selective/unselective template

Theoretical analyses

- fairly good algorithm
- is a better one possible?
- how good will this algorithm be on future data sets
- a lower bound (skipped)
- association rules on random data sets (skipped)
- 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

Chapter 3: Alarm correlation

Part II. Episodes in sequences

3. Alarm correlation: networks and alarms

- network elements: switches, base stations, transmission equipment, etc.
- 10–1000 elements in a network
- an alarm: a message generated by a network element

  `1234 EL1 BTS 940926 082623 A1 Channel missing`
- hundreds of different alarm types
- 200 – 10000 alarms a day
- each contains only local information

Characteristics of the alarm flow

- a variety of situations
- bursts of alarms
- hardware and software change fast

Alarm correlation

"correlating" alarms: combining the fragmented information contained in the alarm sequence and interpreting the whole flow of alarms

- removing redundant alarms
- filtering out low-priority alarms
- replacing alarms by something else
- systems exist
  - knowledge base (correlation rules) constructed manually
  - look at the alarms occurring in a given time window
  - apply actions given in the matching correlation rules

Problem

- how to obtain the information needed for the preparation of an alarm correlation system
- more generally: how to obtain insight into the behavior of the network (alarms)

Solutions

- how to analyze a flow of alarms?
- lots of possibilities: hazard models, neural networks, rule-based representations
- comprehensibility of the discovered knowledge
- simple rule-based representations
- "if certain alarms occur within a time window, then a certain alarm will also occur"

Episodes

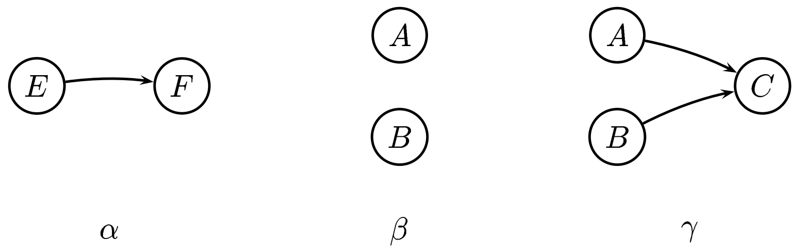


Figure 3.2: Episodes

Basic solution

- look for repeated occurrences of episodes in the alarm flow sequences
- occurrence: alarms of the specified type occur in the specified order
- why this form?
  - comprehensible
  - "standard" for correlation systems
  - represent simple causal relationships
  - insensitive to inaccurate clocks
  - allows analysis of merged, unrelated sequences

Chapter 4: Episodes

4. Frequent episodes

- The framework
- Algorithms
- Experiments

Example sequence

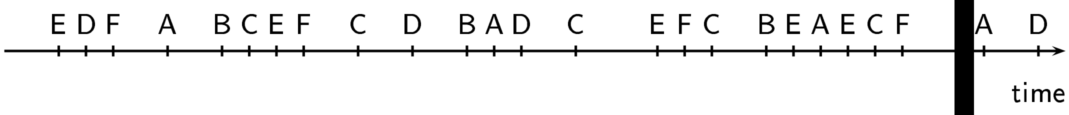


Figure 3.1: A sequence of alarms

Observations:

- whenever E occurs, F occurs soon
- whenever A and B occur (in either order), C occurs soon

Data

- a set R of *event types*
- an *event* is a pair (A, t)
- $A \in R$ is an event type
- t is an integer, the *(occurrence) time* of the event
- *event sequence* s on R: a triple (s, T_s, T_e)
- $T_s < T_e$ are integer (starting and ending time)
- $s = \langle (A_1, t_1), (A_2, t_2), \ldots, (A_n, t_n) \rangle$
- $A_i \in R$ and $T_s \leq t_i < T_e$ for all $i = 1, \ldots, n$
- $t_i \leq t_{i+1}$ for all $i = 1, \ldots, n-1$

Example

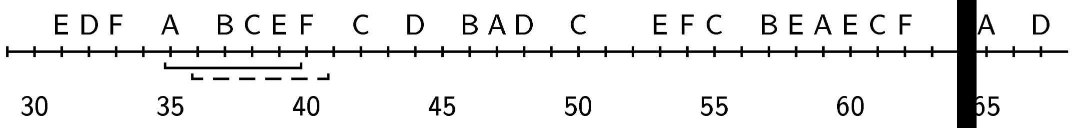


Figure 4.1: The example event sequence s and two windows of width 5.

Windows

- event sequence $\mathbf{s} = (s, T_s, T_e)$
- a window on it: $\mathbf{w} = (w, t_s, t_e)$
- $t_s < T_e, t_e > T_s$
- w consists of those pairs (A, t) from s where $t_s \leq t < t_e$
- $width(\mathbf{w}) = t_e - t_s$: the *width* of the window $\mathbf{w}$
- $\mathcal{W}(\mathbf{s}, win)$: all windows $\mathbf{w}$ on $\mathbf{s}$ such that $width(\mathbf{w}) = win$
- first and last windows!

Episodes

- an *episode* α is a triple $(V, \leq, g)$
- V is a set of nodes
- $\leq$ is a partial order on V
- $g : V \rightarrow R$ is a mapping associating each node with an event type
- intuition: the events in $g(V)$ have to occur in the order described by $\leq$
- *size* of α, denoted $|\alpha|$, is $|V|$
- *parallel episode*: the partial order $\leq$ is trivial
- *serial episode*: $\leq$ is a total order
- *injective*: no event type occurs twice in the episode

Example

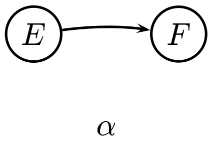


Figure 4.2: An episode

the set V, the mapping g

Example, subepisode

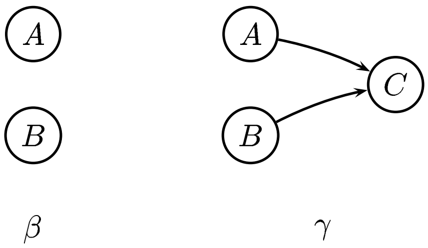


Figure 4.3: A subepisode and episode

Subepisodes

$\beta = (V', \leq', g')$ is a *subepisode* of $\alpha = (V, \leq, g)$, $\beta \preceq \alpha$, if:

there exists an injective mapping $f : V' \to V$ such that

- $g'(v) = g(f(v))$ for all $v \in V'$
- for all $v, w \in V'$ with $v \leq' w$ also $f(v) \leq f(w)$

An episode α is a *superepisode* of β if and only if $\beta \preceq \alpha$

$\beta \prec \alpha$ if $\beta \preceq \alpha$ and $\alpha \not\preceq \beta$

In the example: $\beta \preceq \gamma$

Occurrences of episodes

$\alpha = (V, \leq, g)$ *occurs* in an event sequence $\mathbf{s} = (\langle (A_1, t_1), (A_2, t_2), \ldots, (A_n, t_n) \rangle, T_s, T_e)$, if there exists an injective mapping $h : V \to \{1, \ldots, n\}$ from nodes to events, such that

- $g(x) = A_{h(x)}$ for all $x \in V$
- for all $x, y \in V$ with $x \neq y$ and $x \leq y$ we have $t_{h(x)} < t_{h(y)}$ (or $h(x) < h(y)$)

$(w, 35, 40)$ on the example sequence: events of types A, B, C, and E

both β and γ occur

Frequency of occurrence

- the *frequency* of an episode α in $\mathbf{s}$ is

$$\mathit{fr}(\alpha, \mathbf{s}, \mathit{win}) = \frac{|\{\mathbf{w} \in \mathcal{W}(\mathbf{s}, \mathit{win}) \mid \alpha \text{ occurs in } \mathbf{w}\}|}{|\mathcal{W}(\mathbf{s}, \mathit{win})|},$$

- i.e., the fraction of windows on $\mathbf{s}$ in which α occurs.
- a *frequency threshold min_fr*
- α is *frequent* if $\mathit{fr}(\alpha, \mathbf{s}, \mathit{win}) \geq \mathit{min\_fr}$
- $\mathcal{F}(\mathbf{s}, \mathit{win}, \mathit{min\_fr})$: collection of frequent episodes in $\mathbf{s}$ with respect to *win* and *min_fr*
- size $= l$: $\mathcal{F}_l(\mathbf{s}, \mathit{win}, \mathit{min\_fr})$.

Pattern discovery task

given an event sequence $\mathbf{s}$, a set $\mathcal{E}$ of episodes, a window width *win*, and a frequency threshold *min_fr*, find $\mathcal{F}(\mathbf{s}, \mathit{win}, \mathit{min\_fr})$

Algorithms

Algorithm 4.13

Input: A set R of event types, an event sequence $\mathbf{s}$ over R, a set $\mathcal{E}$ of episodes, a window width *win*, and a frequency threshold *min_fr*.

Output: The collection $\mathcal{F}(\mathbf{s}, \mathit{win}, \mathit{min\_fr})$ of frequent episodes.

Method:
1. compute $\mathcal{C}_1 := \{\alpha \in \mathcal{E} \mid |\alpha| = 1\}$;
2. $l := 1$;
3. **while** $\mathcal{C}_l \neq \emptyset$ **do**
4. // Database pass (Algorithms 4.19 and 4.21):
5. compute $\mathcal{F}_l(\mathbf{s}, \mathit{win}, \mathit{min\_fr}) := \{\alpha \in \mathcal{C}_l \mid \mathit{fr}(\alpha, \mathbf{s}, \mathit{win}) \geq \mathit{min\_fr}\}$;
6. $l := l + 1$;
7. // Candidate generation (Algorithm 4.14):
8. compute $\mathcal{C}_l := \{\alpha \in \mathcal{E} \mid |\alpha| = l,$ and $\beta \in \mathcal{F}_{|\beta|}(\mathbf{s}, \mathit{win}, \mathit{min\_fr})$ for all $\beta \in \mathcal{E}$ such that $\beta \prec \alpha$ and $|\beta| < l\}$;
9. **for** all l **do** output $\mathcal{F}_l(\mathbf{s}, \mathit{win}, \mathit{min\_fr})$;

Basic lemma, once again

Lemma 4.12 If an episode α is frequent in an event sequence $\mathbf{s}$, then all subepisodes $\beta \preceq \alpha$ are frequent. $\square$

Parallel, serial, injective episodes

- *parallel episode*: the partial order $\leq$ is trivial (= frequent sets)
- *serial episode*: $\leq$ is a total order (= frequent subsequence)
- *injective*: no event type occurs twice in the episode (= proper sets, not multi sets)
- useful cases: (serial or parallel) [injective] episodes
  - reduce redundancy in generated episodes
  - keep episodes comprehensible
  - simpler to implement

Generation of candidate episodes

- parallel episodes, serial episodes (injective or non-injective)
- same idea as for association rules
- a candidate episode has to be a combination of two episodes of smaller size
- very small variations to the candidate generation procedure

Recognizing episodes in sequences

- first problem: given a sequence and an episode, find out whether the episode occurs in the sequence
- finding the number of windows containing an occurrence of the episode can be reduced to this
- successive windows have a lot in common
- how to use this?
- an incremental algorithm

Parallel episodes

- for each candidate α maintain a counter $\alpha.$*event_count*: how many events of α are present in the window
- When $\alpha.$*event_count* becomes equal to $|\alpha|$, indicating that α is entirely included in the window
  - save the starting time of the window in $\alpha.$*inwindow*
- when $\alpha.$*event_count* decreases again, increase the field $\alpha.$*freq_count* by the number of windows where α remained entirely in the window

Serial episodes

- use state automata that accept the candidate episodes
- example: episode A B A B

General episodes

different alternatives

Window width (s)	Serial episodes Count	Serial episodes Time (s)	Injective parallel episodes Count	Injective parallel episodes Time (s)
10	16	31	10	8
20	31	63	17	9
40	57	117	33	14
60	87	186	56	15
80	145	271	95	21
100	245	372	139	21
120	359	478	189	22

Table 4.1: Results of experiments with s_1 using a fixed frequency threshold of 0.003 and a varying window width

Frequency threshold	Serial episodes Count	Serial episodes Time (s)	Injective parallel episodes Count	Injective parallel episodes Time (s)
0.1	0	7	0	5
0.05	1	12	1	5
0.008	30	62	19	14
0.004	60	100	40	15
0.002	150	407	93	22
0.001	357	490	185	22

Table 4.2: Results of experiments with $\mathbf{s}_1$ using a fixed window width of 60 s and a varying frequency threshold

Episode size	Number of episodes	Number of candidate episodes	Number of frequent episodes	Match
1	287	287.0	30.1	11 %
2	82 369	1 078.7	44.6	4 %
3	$2 \cdot 10^{7}$	192.4	20.0	10 %
4	$7 \cdot 10^{9}$	17.4	10.1	58 %
5	$2 \cdot 10^{12}$	7.1	5.3	74 %
6	$6 \cdot 10^{14}$	4.7	2.9	61 %
7	$2 \cdot 10^{17}$	2.9	2.1	75 %
8	$5 \cdot 10^{19}$	2.1	1.7	80 %
9	$1 \cdot 10^{22}$	1.7	1.4	83 %
10–		17.4	16.0	92 %

Table 4.3: Number of candidate and frequent serial episodes in $\mathbf{s}_1$ with frequency threshold 0.003 and averaged over window widths 10, 20, 40, 60, 80, 100, and 120 s

Experiences in alarm correlation

Useful in

- finding long-term, rather frequently occurring dependencies,
- creating an overview of a short-term alarm sequence, and
- evaluating the consistency and correctness of alarm databases
- discovered rules have been applied in alarm correlation
- lots of rules are trivial

Chapter 5: Minimal occurrences of episodes

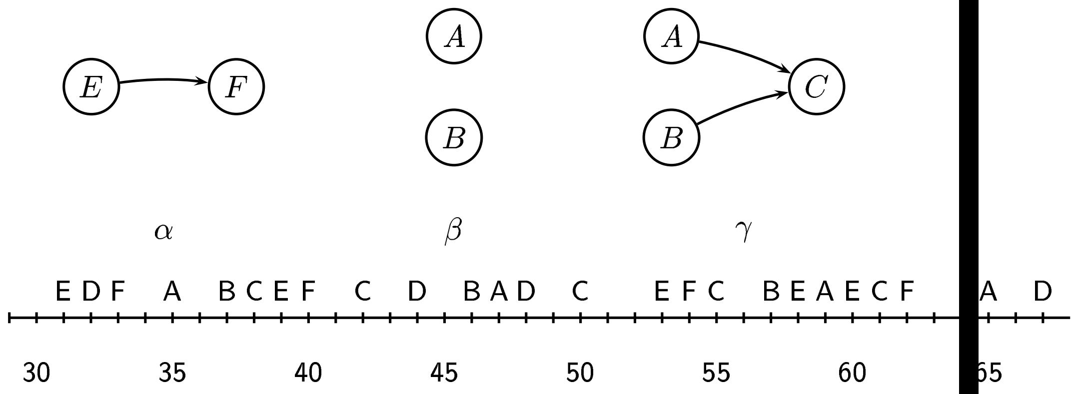
5. Minimal occurrences of episodes

- an alternative approach to discovery of episodes
- no windows
- for each potentially interesting episode, find out the exact occurrences of the episode
- advantages: easy to modify time limits, several time limits for one rule ("if A and B occur within 15 seconds, then C follows within 30 seconds")
- disadvantages: uses lots of space

Definitions

- an episode α and an event sequence $\mathbf{s}$
- interval $[t_s, t_e)$ is a *minimal occurrence* of α in $\mathbf{s}$, if
  - α occurs in the window $\mathbf{w} = (w, t_s, t_e)$ on $\mathbf{s}$
  - α does not occur in any proper subwindow on $\mathbf{w}$
- *set of (intervals of) minimal occurrences* of an episode α: *mo*$(\alpha) = \{ [t_s, t_e) \mid [t_s, t_e)$ is a minimal occurrence of $\alpha\}$.

Example



Figures 5.1: Episodes and 5.2: The example event sequence s

$mo(\beta) = \{[35, 38), [46, 48), [47, 58), [57, 60)\}$.

$mo(\gamma) = \{[35, 39), [46, 51), [57, 62)\}$.

Episodes rules, new version

- *episode rule*: $\beta\,[win_1] \;\Rightarrow\; \alpha\,[win_2]$,
- β and α are episodes such that $\beta \preceq \alpha$
- win_1 and win_2 are integers
- if episode β has a minimal occurrence at interval $[t_s, t_e)$ with $t_e - t_s \le win_1$, then episode α occurs at interval $[t_s, t'_e)$ for some t'_e such that $t'_e - t_s \le win_2$
- (old version: $\beta\,[w] \;\Rightarrow\; \alpha\,[w]$, in windows containing β)

- formally: $mo_{win_1}(\beta) = \{[t_s, t_e) \in mo(\beta) \mid t_e - t_s \leq win_1\}$
- given α and an interval $[u_s, u_e)$, define $occ(\alpha, [u_s, u_e)) =$ true if and only if there exists a minimal occurrence $[u'_s, u'_e) \in mo(\alpha)$ such that $u_s \leq u'_s$ and $u'_e \leq u_e$
- The confidence of an episode rule $\beta\,[win_1] \Rightarrow \alpha\,[win_2]$ is now

$$\frac{|\{[t_s, t_e) \in mo_{win_1}(\beta) \mid occ(\alpha, [t_s, t_s + win_2))\}|}{|mo_{win_1}(\beta)|}.$$

Example, cont.

- $\beta\,[3] \Rightarrow \gamma\,[4]$
- three minimal occurrences $[35, 38), [46, 48), [57, 60)$ of β of width at most 3 in the denominator
- Only $[35, 38)$, has an occurrence of α within width 4, so the confidence is $1/3$.
- rule $\beta\,[3] \Rightarrow \gamma\,[5]$ the confidence is 1.

Rule forms

- temporal relationships can be complex

Frequency and support

- previously: frequency = fraction of windows containing the episode
- no fixed window size
- several minimal occurrences within a window
- support of an episode: the number of minimal occurrences of an episode, $|mo(\alpha)|$

Rule discovery task

- an event sequence $\mathbf{s}$
- a frequency threshold *min_fr*
- a class $\mathcal{E}$ of episodes
- a set W of time bounds
- find all frequent episode rules of the form $\beta\,[\mathit{win}_1] \Rightarrow \alpha\,[\mathit{win}_2]$
- $\beta, \alpha \in \mathcal{E}$ and $\mathit{win}_1, \mathit{win}_2 \in W$.

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
   ⇒ 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

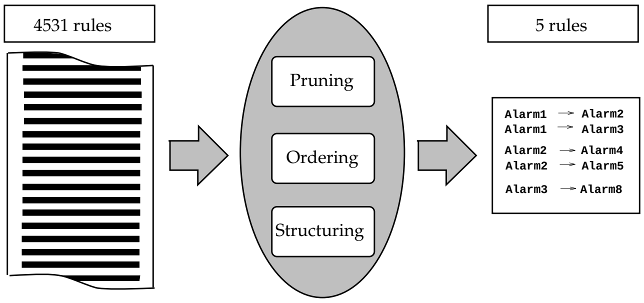
Presentation and analysis of results

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



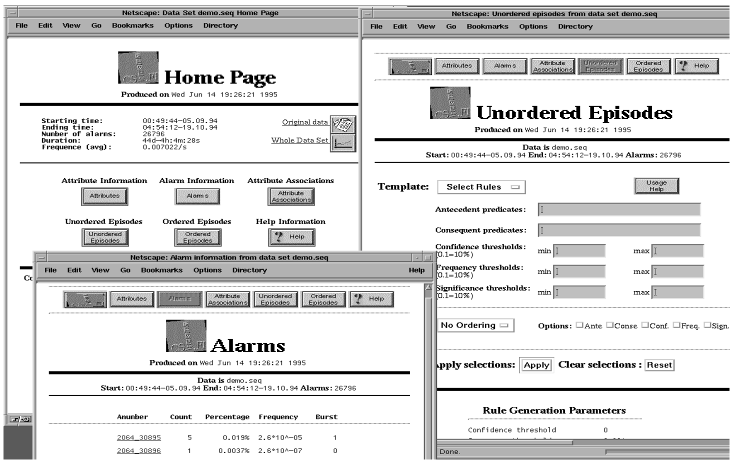
Pruning and ordering:

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

Structuring:

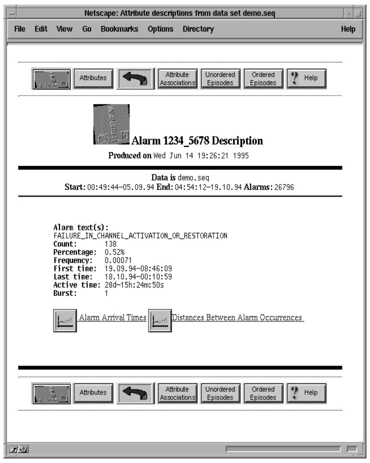
- clusters, hierarchies, etc.

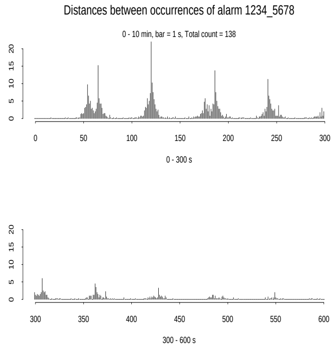
TASA: A KDD tool for alarm analysis



- pages are created automatically from analysis results

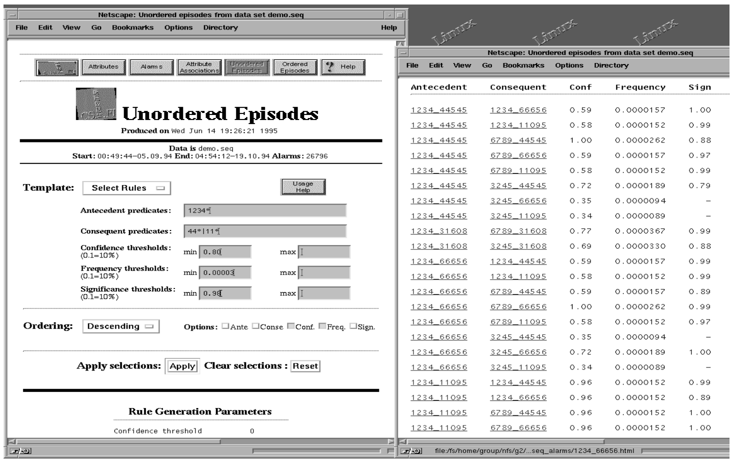
TASA: Giving an overview of data





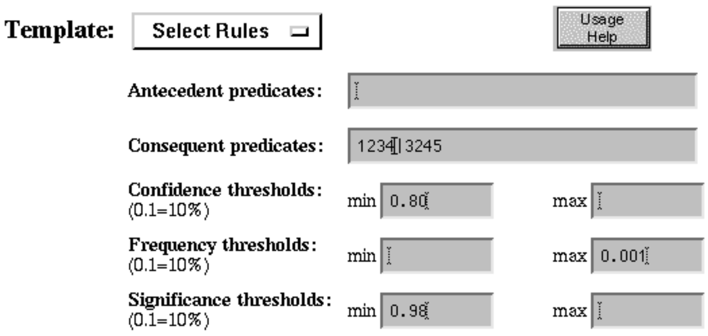
statistical information, histograms

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 $fr(\varphi, r) \geq min\_fr$

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 C_1 := {φ ∈ P | level(φ) = 1};
2.  l := 1;
3.  while C_l ≠ ∅ do
4.      // Database pass:
5.      compute T_l(P, r, q) := {φ ∈ C_l | q(φ, r)};
6.      l := l + 1;
7.      // Candidate generation:
8.      compute C_l := {φ ∈ P | level(φ) = l and θ ∈ T_level(θ)(P, r, q) for all
                                                    θ ∈ P such that θ ≺ φ};
9.  for all l do output T_l(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

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 $\mathcal{P}$
- t 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

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

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

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

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

Example

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

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

- 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 $A, \ldots, D$

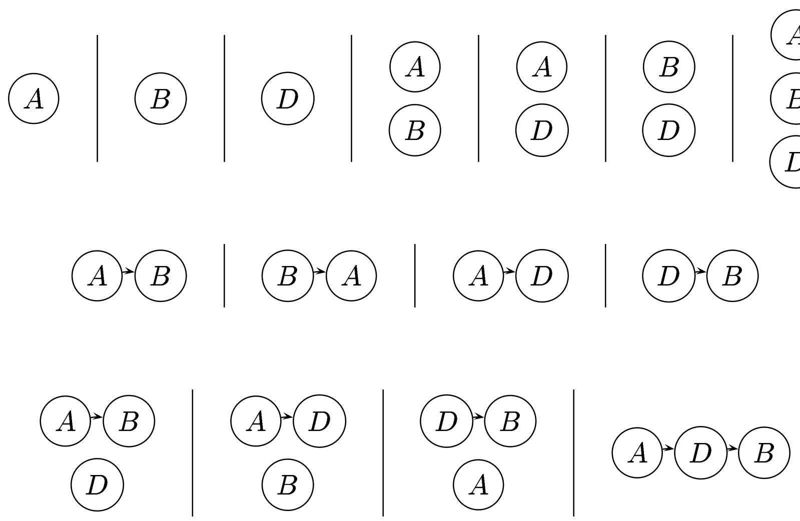


Figure 8.1: A collection $\mathcal{F}(\mathrm{s}, \mathit{win}, \mathit{min\_fr})$ of frequent episodes.

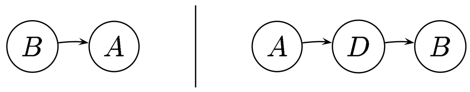


Figure 8.2: The positive border $\mathcal{B}d^+(\mathcal{F}(\mathrm{s}, \mathit{win}, \mathit{min\_fr}))$.

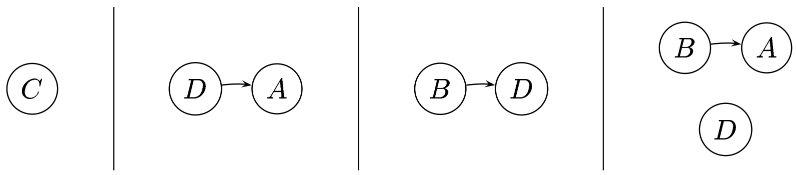


Figure 8.3: The negative border $\mathcal{B}d^-(\mathcal{F}(\mathrm{s}, \mathit{win}, \mathit{min\_fr}))$.

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 $\mathcal{T}(\mathcal{P}, \mathbf{r}, q) \cup \mathcal{B}d^-(\mathcal{T}(\mathcal{P}, \mathbf{r}, q))$. $\square$

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 $\mathcal{F}(r, \mathit{min\_fr}) \cup \mathcal{B}d^-(\mathcal{F}(r, \mathit{min\_fr}))$. $\square$

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:

$\mathcal{C}^* := \emptyset$;

// correct $\mathcal{S}$ downward:

$\mathcal{C} := \mathcal{B}d^+(\mathcal{S})$;

while $\mathcal{C} \neq \emptyset$ **do**

  $\mathcal{C}^* := \mathcal{C}^* \cup \mathcal{C}$;

  $\mathcal{S} := \mathcal{S} \setminus \{\varphi \in \mathcal{C} \mid q(\mathbf{r}, \varphi)$ is false$\}$;

  $\mathcal{C} := \mathcal{B}d^+(\mathcal{S}) \setminus \mathcal{C}^*$;

od;

// now $\mathcal{S} \subseteq \mathcal{T}(\mathcal{P}, \mathbf{r}, q)$; expand $\mathcal{S}$ upwards:

$\mathcal{C} := \mathcal{B}d^-(\mathcal{S}) \setminus \mathcal{C}^*$;

while $\mathcal{C} \neq \emptyset$ **do**

  $\mathcal{C}^* := \mathcal{C}^* \cup \mathcal{C}$;

  $\mathcal{S} := \mathcal{S} \cup \{\varphi \in \mathcal{C} \mid q(\mathbf{r}, \varphi)$ is true$\}$;

  $\mathcal{C} := \mathcal{B}d^-(\mathcal{S}) \setminus \mathcal{C}^*$;

od;

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. □

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 $\mathcal{B}d(\mathcal{T}(\mathcal{P}, \mathbf{r}, q))$. □

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

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; the *complements* of the sets in the positive border be the edges of a simple hypergraph $\mathcal{H}$.
- 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
  $\Rightarrow$ 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

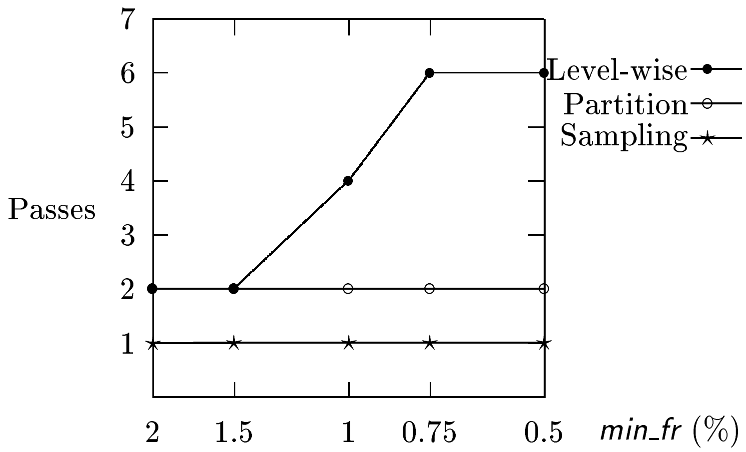


Figure 9.1 The number of database passes for frequent set algorithms (T10.I4.D100K)

Results

Lowered frequency threshold

min_fr (%)	Sample size $\lvert s \rvert$ 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

min_fr	Sample size 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