Temporal Data Mining

Christian Moewes

Otto-von-Guericke University of Magdeburg
Faculty of Computer Science
Department of Knowledge Processing and Language Engineering
Outline

1. Introduction
   Data Mining
   Knowledge discovery in databases
   CRISP-DM
   Why to Study Temporal Data Mining?

2. Association Rules and Frequent Item Sets

3. Frequent Sequence Mining

4. Finding Motifs in Time Series Effectively
Data

- today: companies/institutes maintain huge databases
- gigantic archives of tables, documents, images, sounds
- “If you have enough data, you can solve any problem!”
- in large databases: can’t see the wood for the trees
- patterns, structures, regularities stay undetected
- finding patterns and exploit information is fairly difficult

*We are drowning in information but starved for knowledge.*

[John Naisbitt]
Knowledge discovery in databases

- actually, abundance of data
- lack of tools transforming data into knowledge

⇒ research area: knowledge discovery in databases (KDD)
- nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data
- one step in KDD: data mining

Miner VGA (1989) screenshot
Data mining tasks

- classification
  *Is this customer creditworthy or not?*

- segmentation, clustering
  *What groups of customers do I have?*

- concept description
  *Which properties characterize cured patients?*

- prediction
  *How many new cars will be called next month?*

- dependence/association analysis
  *Which CAN-bus errors of broken cars occur together frequently?*

- ...
Temporal Data Mining

- many data mining problems deal with temporal features
- most prominent: (numeric/symbolic) time series
- time series are ubiquitous: finance, medicine, biometry, chemistry, astronomy, robotics, networks and industry
- upcoming: temporal sequences of arbitrary objects, e.g. subgraphs
- challenges: preprocessing, (dis)similarity measures, representation, search for useful information
- last decade: research on purely proposing new algorithms
- nowadays: research on application-based solutions
Outline

1. Introduction

2. Association Rules and Frequent Item Sets
   - The Apriori Algorithm
   - Association Rules
   - Rule Icons: Industrial Applications

3. Frequent Sequence Mining

4. Finding Motifs in Time Series Effectively
Frequent Item Set Mining: Motivation

- method for **market basket analysis**
- finding regularities in shopping behavior of customers of supermarkets, mail-order companies, on-line shops etc.
- more specifically: **find sets of products that are frequently bought together**
- possible applications:
  - improve arrangement of products in shelves, on catalog’s pages
  - support cross-selling (suggestion of other products), product bundling
  - fraud detection, technical dependence analysis
- often found patterns are expressed as **association rules**, *e.g.*
  
  *If a customer buys bread and wine, then she/he will probably also buy cheese.*
Frequent Item Set Mining: Basic Notions

- let $A = \{a_1, \ldots, a_m\}$ be set of **items** (products, special equipment items, service options, ...)

- any subset $I \subseteq A$ is called **item set**
  item set: set of products that can be bought (together)

- let $T = (t_1, \ldots, t_n)$ with $\forall i, 1 \leq i \leq n : t_i \subseteq A$ be vector of **transactions** over $A$,
  each transaction is item set, but some item sets may $\notin T$
  transactions needn’t be pairwise different: may be $t_i = t_k$ for $i \neq k$
  $T$ may also be **bag** or **multiset** of transactions
  $A$ may not be explicitly given, but only implicitly as $A = \bigcup_{i=1}^{n} t_i$
  $T$ can list, e.g. sets of products bought by customers of supermarket in given period of time
Frequent Item Set Mining: Basic Notions

let \( I \subseteq A \) be item set and \( T \) vector of transactions over \( A \)

- transaction \( t \in T \) covers item set \( I \) or item set \( I \) is contained in transaction \( t \in T \) iff \( I \subseteq t \).
- set \( K_T(I) = \{k \in \{1, \ldots, n\} \mid I \subseteq t_k\} \) is called cover of \( I \) w.r.t. \( T \) cover of item set is index set of transactions that cover it it may also be defined as vector of all transactions that cover it (however, this is complicated to write in formally correct way)
- value \( s_T(I) = |K_T(I)| \) is called (absolute) support of \( I \) w.r.t. \( T \) value \( \sigma_T(I) = \frac{1}{n} |K_T(I)| \) is called relative support of \( I \) w.r.t. \( T \) support of \( I \) is number or fraction of transactions that contain it sometimes \( \sigma_T(I) \) is also called (relative) frequency of \( I \) w.r.t. \( T \)
Frequent Item Set Mining: Formal Definition

given:

- set $A = \{a_1, \ldots, a_m\}$ of items,
- vector $T = (t_1, \ldots, t_n)$ of transactions over $A$,
- number $s_{\text{min}} \in \mathbb{N}$, $0 < s_{\text{min}} \leq n$ or (equivalently) number $\sigma_{\text{min}} \in \mathbb{R}$, $0 < \sigma_{\text{min}} \leq 1$, the minimum support desired:

- set of frequent item sets, i.e.
  set $F_T(s_{\text{min}}) = \{I \subseteq A \mid s_T(I) \geq s_{\text{min}}\}$ or (equivalently)
  set $\Phi_T(\sigma_{\text{min}}) = \{I \subseteq A \mid \sigma_T(I) \geq \sigma_{\text{min}}\}$

note: with the relations $s_{\text{min}} = \lceil n\sigma_{\text{min}} \rceil$ and $\sigma_{\text{min}} = \frac{1}{n}s_{\text{min}}$ the two versions can easily be transformed into each other
Frequent Item Sets: Example

transaction vector

1: \{a, d, e\}
2: \{b, c, d\}
3: \{a, c, e\}
4: \{a, c, d, e\}
5: \{a, e\}
6: \{a, c, d\}
7: \{b, c\}
8: \{a, c, d, e\}
9: \{c, b, e\}
10: \{a, d, e\}

frequent item sets

<table>
<thead>
<tr>
<th>0 items</th>
<th>1 item</th>
<th>2 items</th>
<th>3 items</th>
</tr>
</thead>
<tbody>
<tr>
<td>\emptyset: 1</td>
<td>{a}: .7</td>
<td>{a, c}: .4</td>
<td>{a, c, d}: .3</td>
</tr>
<tr>
<td></td>
<td>{b}: .3</td>
<td>{a, d}: .5</td>
<td>{a, c, e}: .3</td>
</tr>
<tr>
<td></td>
<td>{c}: .7</td>
<td>{a, e}: .6</td>
<td>{a, d, e}: .4</td>
</tr>
<tr>
<td></td>
<td>{d}: .6</td>
<td>{b, c}: .3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{e}: .7</td>
<td>{c, d}: .4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{c, e}: .4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{d, e}: .4</td>
<td></td>
</tr>
</tbody>
</table>

- minimum support \( s_{\text{min}} = 3 \) or \( \sigma_{\text{min}} = 0.3 = 30\% \) in this example
- there are \( ??? \) possible item sets over \( A = \{a, b, c, d, e\} \)
- there are 16 frequent item sets (but only 10 transactions)
Frequent Item Sets: Example

transaction vector

1: \{a, d, e\}
2: \{b, c, d\}
3: \{a, c, e\}
4: \{a, c, d, e\}
5: \{a, e\}
6: \{a, c, d\}
7: \{b, c\}
8: \{a, c, d, e\}
9: \{c, b, e\}
10: \{a, d, e\}

frequent item sets

<table>
<thead>
<tr>
<th>0 items</th>
<th>1 item</th>
<th>2 items</th>
<th>3 items</th>
</tr>
</thead>
<tbody>
<tr>
<td>\emptyset: 1</td>
<td>{a}: .7</td>
<td>{a, c}: .4</td>
<td>{a, c, d}: .3</td>
</tr>
<tr>
<td></td>
<td>{b}: .3</td>
<td>{a, d}: .5</td>
<td>{a, c, e}: .3</td>
</tr>
<tr>
<td></td>
<td>{c}: .7</td>
<td>{a, e}: .6</td>
<td>{a, d, e}: .4</td>
</tr>
<tr>
<td></td>
<td>{d}: .6</td>
<td>{b, c}: .3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{e}: .7</td>
<td>{c, d}: .4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{c, e}: .4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{d, e}: .4</td>
<td></td>
</tr>
</tbody>
</table>

- minimum support $s_{\text{min}} = 3$ or $\sigma_{\text{min}} = 0.3 = 30\%$ in this example
- there are $2^5 = 32$ possible item sets over $A = \{a, b, c, d, e\}$
- there are 16 frequent item sets (but only 10 transactions)
Properties of the Support of an Item Set

- **brute force approach** (enumerate all possible item sets, determine their support and discard infrequent item sets) is **infeasible**: number of possible item sets grows exponentially with number of items
typical supermarket has thousands of different products

- **idea**: consider properties of support, in particular:

  \[
  \forall I : \forall J \supseteq I : \quad K_T(J) \subseteq K_T(I)
  \]

  this property holds, since

  \[
  \forall t : \forall I : \forall J \supseteq I : \quad J \subseteq t \rightarrow I \subseteq t
  \]

  each item is one condition a transaction must satisfy
transactions not satisfying this condition are removed from cover

- it follows: \[
  \forall I : \forall J \supseteq I : \quad s_T(I) \geq s_T(J) \]
i.e.
if an item set is extended, its support cannot increase
support is **anti-monotone** or **downward closed**
Properties of the Support of an Item Set

• from \( \forall I : \forall J \supseteq I : s_T(I) \geq s_T(J) \) it follows

\[ \forall s_{\text{min}} : \forall I : \forall J \supseteq I : s_T(I) < s_{\text{min}} \rightarrow s_T(J) < s_{\text{min}}. \]

i.e. no superset of an infrequent item set can be frequent

• Apriori property
  rationale: sometimes we can know a priori, i.e. before checking its support by accessing given transaction vector, that an item set cannot be frequent

• contraposition of this implication also holds:

\[ \forall s_{\text{min}} : \forall I : \forall J \subseteq I : s_T(I) \geq s_{\text{min}} \rightarrow s_T(J) \geq s_{\text{min}}. \]

i.e. all subsets of frequent item set are frequent

• compressed representation of set of frequent item sets
Maximal Item Sets

• consider set of maximal (frequent) item sets:

\[
M_T(s_{\text{min}}) = \{ I \subseteq A \mid s_T(I) \geq s_{\text{min}} \land \forall J \supset I : s_T(J) < s_{\text{min}} \}
\]

i.e. item set is maximal if it is frequent, but none of its proper supersets is frequent

• so, we know that

\[
\forall s_{\text{min}} : \forall I : \quad I \in M_T(s_{\text{min}}) \lor \exists J \supset I : s_T(J) \geq s_{\text{min}}
\]

it follows

\[
\forall s_{\text{min}} : \forall I : \quad I \in F_T(s_{\text{min}}) \rightarrow \exists J \in M_T(s_{\text{min}}) : I \subseteq J
\]

i.e. every frequent item set has a maximal superset

• therefore:

\[
\forall s_{\text{min}} : \quad F_T(s_{\text{min}}) = \bigcup_{I \in M_T(s_{\text{min}})} 2^I
\]
Maximal Frequent Item Sets: Example

transaction vector

1: \{a, d, e\}
2: \{b, c, d\}
3: \{a, c, e\}
4: \{a, c, d, e\}
5: \{a, e\}
6: \{a, c, d\}
7: \{b, c\}
8: \{a, c, d, e\}
9: \{c, b, e\}
10: \{a, d, e\}

frequent item sets

<table>
<thead>
<tr>
<th>0 items</th>
<th>1 item</th>
<th>2 items</th>
<th>3 items</th>
</tr>
</thead>
<tbody>
<tr>
<td>\emptyset: 1</td>
<td>{a}: .7</td>
<td>{a, c}: .4</td>
<td>{a, c, d}: .3</td>
</tr>
<tr>
<td></td>
<td>{b}: .3</td>
<td>{a, d}: .5</td>
<td>{a, c, e}: .3</td>
</tr>
<tr>
<td></td>
<td>{c}: .7</td>
<td>{a, e}: .6</td>
<td>{a, d, e}: .4</td>
</tr>
<tr>
<td></td>
<td>{d}: .6</td>
<td>{b, c}: .3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{e}: .7</td>
<td>{c, d}: .4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{c, e}: .4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{d, e}: .4</td>
<td></td>
</tr>
</tbody>
</table>

- which item sets are maximal?

- every frequent item set is a subset of at least one of these sets
Maximal Frequent Item Sets: Example

transaction vector

1: \{a, d, e\}
2: \{b, c, d\}
3: \{a, c, e\}
4: \{a, c, d, e\}
5: \{a, e\}
6: \{a, c, d\}
7: \{b, c\}
8: \{a, c, d, e\}
9: \{c, b, e\}
10: \{a, d, e\}

frequent item sets

<table>
<thead>
<tr>
<th>0 items</th>
<th>1 item</th>
<th>2 items</th>
<th>3 items</th>
</tr>
</thead>
<tbody>
<tr>
<td>{} : 1</td>
<td>{a} : .7</td>
<td>{a, c} : .4</td>
<td>{a, c, d} : .3</td>
</tr>
<tr>
<td></td>
<td>{b} : .3</td>
<td>{a, d} : .5</td>
<td>{a, c, e} : .3</td>
</tr>
<tr>
<td></td>
<td>{c} : .7</td>
<td>{a, e} : .6</td>
<td>{a, d, e} : .4</td>
</tr>
<tr>
<td></td>
<td>{d} : .6</td>
<td>{b, c} : .3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{e} : .7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- which item sets are maximal?
  - \{b, c\},

- every frequent item set is a subset of at least one of these sets
Maximal Frequent Item Sets: Example

transaction vector

1: \{a, d, e\}
2: \{b, c, d\}
3: \{a, c, e\}
4: \{a, c, d, e\}
5: \{a, e\}
6: \{a, c, d\}
7: \{b, c\}
8: \{a, c, d, e\}
9: \{c, b, e\}
10: \{a, d, e\}

frequent item sets

<table>
<thead>
<tr>
<th>0 items</th>
<th>1 item</th>
<th>2 items</th>
<th>3 items</th>
</tr>
</thead>
<tbody>
<tr>
<td>\emptyset: 1</td>
<td>{a}: .7</td>
<td>{a, c}: .4</td>
<td>{a, c, d}: .3</td>
</tr>
<tr>
<td></td>
<td>{b}: .3</td>
<td>{a, d}: .5</td>
<td>{a, c, e}: .3</td>
</tr>
<tr>
<td></td>
<td>{c}: .7</td>
<td>{a, e}: .6</td>
<td>{a, d, e}: .4</td>
</tr>
<tr>
<td></td>
<td>{d}: .6</td>
<td>{b, c}: .3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{e}: .7</td>
<td>{c, d}: .4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{c, e}: .4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{d, e}: .4</td>
<td></td>
</tr>
</tbody>
</table>

- which item sets are maximal?
  \{b, c\}, \{a, c, d\},

- every frequent item set is a subset of at least one of these sets
Maximal Frequent Item Sets: Example

transaction vector

1: \{a, d, e\}
2: \{b, c, d\}
3: \{a, c, e\}
4: \{a, c, d, e\}
5: \{a, e\}
6: \{a, c, d\}
7: \{b, c\}
8: \{a, c, d, e\}
9: \{c, b, e\}
10: \{a, d, e\}

frequent item sets

<table>
<thead>
<tr>
<th>0 items</th>
<th>1 item</th>
<th>2 items</th>
<th>3 items</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: 1</td>
<td>{a}: .7</td>
<td>{a, c}: .4</td>
<td>{a, c, d}: .3</td>
</tr>
<tr>
<td></td>
<td>{b}: .3</td>
<td>{a, d}: .5</td>
<td>{a, c, e}: .3</td>
</tr>
<tr>
<td></td>
<td>{c}: .7</td>
<td>{a, e}: .6</td>
<td>{a, d, e}: .4</td>
</tr>
<tr>
<td></td>
<td>{d}: .6</td>
<td>{b, c}: .3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{e}: .7</td>
<td>{c, d}: .4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{c, e}: .4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{d, e}: .4</td>
<td></td>
</tr>
</tbody>
</table>

- which item sets are maximal?
  \{b, c\}, \{a, c, d\}, \{a, c, e\},

- every frequent item set is a subset of at least one of these sets
Maximal Frequent Item Sets: Example

transaction vector

1: \{a, d, e\}
2: \{b, c, d\}
3: \{a, c, e\}
4: \{a, c, d, e\}
5: \{a, e\}
6: \{a, c, d\}
7: \{b, c\}
8: \{a, c, d, e\}
9: \{c, b, e\}
10: \{a, d, e\}

frequent item sets

<table>
<thead>
<tr>
<th>0 items</th>
<th>1 item</th>
<th>2 items</th>
<th>3 items</th>
</tr>
</thead>
<tbody>
<tr>
<td>∅: 1</td>
<td>{a}: .7</td>
<td>{a, c}: .4</td>
<td>{a, c, d}: .3</td>
</tr>
<tr>
<td></td>
<td>{b}: .3</td>
<td>{a, d}: .5</td>
<td>{a, c, e}: .3</td>
</tr>
<tr>
<td></td>
<td>{c}: .7</td>
<td>{a, e}: .6</td>
<td>{a, d, e}: .4</td>
</tr>
<tr>
<td></td>
<td>{d}: .6</td>
<td>{b, c}: .3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{e}: .7</td>
<td>{c, d}: .4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{c, e}: .4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{d, e}: .4</td>
<td></td>
</tr>
</tbody>
</table>

- which item sets are maximal?

\{b, c\}, \{a, c, d\}, \{a, c, e\}, \{a, d, e\}

- every frequent item set is a subset of at least one of these sets
Limits of Maximal Item Sets

- set of maximal item sets captures set of all frequent item sets, but then we only know support of maximal item sets
- about support of a non-maximal frequent item set we only know:

\[ \forall s_{\text{min}} : \forall I \in F_T(s_{\text{min}}) - M_T(s_{\text{min}}) : s_T(I) \geq \max_{J \in M_T(s_{\text{min}}), J \supset I} s_T(J) \]

this follows immediately from \( \forall I : \forall J \supset I : s_T(I) \geq s_T(J) \), i.e. an item set cannot have lower support than any of its supersets

- note, in general

\[ \forall s_{\text{min}} : \forall I \in F_T(s_{\text{min}}) : s_T(I) \geq \max_{J \in M_T(s_{\text{min}}), J \supseteq I} s_T(J) \]

- question: can we find subset of set of all frequent item sets, which also preserves knowledge of all support values?
Closed Item Sets

• consider set of **closed (frequent) item sets**:

\[ C_T(s_{\text{min}}) = \{ I \subseteq A \mid s_T(I) \geq s_{\text{min}} \land \forall J \supset I : s_T(J) < s_T(I) \} \]

i.e. item set is closed if it is frequent, but none of its proper supersets has same support

• with this we know that

\[ \forall s_{\text{min}} : \forall I : I \in C_T(s_{\text{min}}) \lor \exists J \supset I : s_T(J) = s_T(I) \]

it follows

\[ \forall s_{\text{min}} : \forall I : I \in F_T(s_{\text{min}}) \rightarrow \exists J \in C_T(s_{\text{min}}) : I \subseteq J \]

i.e. **every frequent item set has a closed superset**

• therefore:

\[ \forall s_{\text{min}} : F_T(s_{\text{min}}) = \bigcup_{I \in C_T(s_{\text{min}})} 2^I \]
Closed Item Sets

• however, not only every frequent item set has closed superset, but it has **closed superset with the same support**:

\[
\forall s_{\text{min}} : \forall I : I \in F_T(s_{\text{min}}) \rightarrow \exists J \supseteq I : J \in C_T(s_{\text{min}}) \land s_T(J) = s_T(I)
\]

(proof: see considerations on next slide)

• set of all closed item sets preserves knowledge of all support values:

\[
\forall s_{\text{min}} : \forall I \in F_T(s_{\text{min}}) : s_T(I) = \max_{J \in C_T(s_{\text{min}}), J \supseteq I} s_T(J)
\]

• note: weaker statement

\[
\forall s_{\text{min}} : \forall I \in F_T(s_{\text{min}}) : s_T(I) \geq \max_{J \in C_T(s_{\text{min}}), J \supseteq I} s_T(J)
\]

follows immediately from \( \forall I : \forall J \supseteq I : s_T(I) \geq s_T(J) \), *i.e.* item set cannot have lower support than any of its supersets
Closed Item Sets

- alternative characterization:

\[ I \text{ closed } \iff s_T(I) \geq s_{\text{min}} \land I = \bigcap_{k \in K_T(I)} t_k. \]

reminder: \( K_T(I) = \{ k \in \{1, \ldots, n\} \mid I \subseteq t_k \} \) is cover of \( I \) w.r.t. \( T \)

- derived as follows: since \( \forall k \in K_T(I) : I \subseteq t_k \), it is obvious that

\[ \forall s_{\text{min}} : \forall I \in F_T(s_{\text{min}}) : I \subseteq \bigcap_{k \in K_T(I)} t_k, \]

if \( I \subset \bigcap_{k \in K_T(I)} t_k \), it is not closed, since \( \bigcap_{k \in K_T(I)} t_k \) has same support

on the other hand, no superset of \( \bigcap_{k \in K_T(I)} t_k \) has cover \( K_T(I) \)

- note: above characterization allows to construct (uniquely determined) closed superset of frequent item set with same support
Closed Frequent Item Sets: Example

transaction vector

1: \{a, d, e\}
2: \{b, c, d\}
3: \{a, c, e\}
4: \{a, c, d, e\}
5: \{a, e\}
6: \{a, c, d\}
7: \{b, c\}
8: \{a, c, d, e\}
9: \{c, b, e\}
10: \{a, d, e\}

frequent item sets

<table>
<thead>
<tr>
<th></th>
<th>0 items</th>
<th>1 item</th>
<th>2 items</th>
<th>3 items</th>
</tr>
</thead>
<tbody>
<tr>
<td>\emptyset</td>
<td>{a}: .7</td>
<td>{a, c}: .4</td>
<td>{a, c, d}: .3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{b}: .3</td>
<td>{a, d}: .5</td>
<td>{a, c, e}: .3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{c}: .7</td>
<td>{a, e}: .6</td>
<td>{a, d, e}: .4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{d}: .6</td>
<td>{b, c}: .3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>{e}: .7</td>
<td>{c, d}: .4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- all frequent item sets but \{b\} and \{d, e\} are closed
- \{b\} is subset of \{b, c\}, both have support 0.3
- \{d, e\} is subset of \{a, d, e\}, both have support 0.4
Types of Frequent Item Sets

- **Frequent Item Set**: any frequent item set (support is higher than minimal support):
  \[ I \text{ frequent } \iff s_T(I) \geq s_{\text{min}} \]

- **Closed Item Set**: frequent item set is called *closed* if no superset has same support:
  \[ I \text{ closed } \iff s_T(I) \geq s_{\text{min}} \land \forall J \supset I : s_T(J) < s_T(I) \]

- **Maximal Item Set**: frequent item set is called *maximal* if no superset is frequent:
  \[ I \text{ maximal } \iff s_T(I) \geq s_{\text{min}} \land \forall J \supset I : s_T(J) < s_{\text{min}} \]

- **obvious relations**
  - all maximal and all closed item sets are frequent
  - all maximal item sets are closed
## Types of Frequent Item Sets: Example

<table>
<thead>
<tr>
<th></th>
<th>0 items</th>
<th>1 item</th>
<th>2 items</th>
<th>3 items</th>
</tr>
</thead>
<tbody>
<tr>
<td>∅</td>
<td>1</td>
<td>{a} .7</td>
<td>{a, c} .4</td>
<td>{a, c, d} .3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{b} .3</td>
<td>{a, d} .5</td>
<td>{a, c, e} .3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{c} .7</td>
<td>{a, e} .6</td>
<td>{a, d, e} .4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{d} .6</td>
<td>{b, c} .3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{e} .7</td>
<td>{c, d} .4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>{c, e} .4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>{d, e} .4</td>
<td></td>
</tr>
</tbody>
</table>

- **Frequent Item Set:** any frequent item set (support is higher than the minimal support)
- **Closed Item Set:** (marked with †) a frequent item set is called *closed* if no superset has the same support
- **Maximal Item Set:** (marked with *) a frequent item set is called *maximal* if no superset is frequent
### Types of Frequent Item Sets: Example

<table>
<thead>
<tr>
<th>0 items</th>
<th>1 item</th>
<th>2 items</th>
<th>3 items</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset^+$ 1</td>
<td>${a}^+ .7$</td>
<td>${a, c}^+ .4$</td>
<td>${a, c, d}^+ .3$</td>
</tr>
<tr>
<td></td>
<td>${b} .3$</td>
<td>${a, d}^+ .5$</td>
<td>${a, c, e}^+ .3$</td>
</tr>
<tr>
<td></td>
<td>${c}^+ .7$</td>
<td>${a, e}^+ .6$</td>
<td>${a, d, e}^+ .4$</td>
</tr>
<tr>
<td></td>
<td>${d}^+ .6$</td>
<td>${b, c}^+ .3$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>${e}^+ .7$</td>
<td>${c, d}^+ .4$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>${c, e}^+ .4$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>${d, e} .4$</td>
<td></td>
</tr>
</tbody>
</table>

- **Frequent Item Set:** any frequent item set (support is higher than the minimal support)
- **Closed Item Set:** (marked with $^+$) a frequent item set is called *closed* if no superset has the same support
- **Maximal Item Set:** (marked with $^*$) a frequent item set is called *maximal* if no superset is frequent
### Types of Frequent Item Sets: Example

<table>
<thead>
<tr>
<th>0 items</th>
<th>1 item</th>
<th>2 items</th>
<th>3 items</th>
</tr>
</thead>
<tbody>
<tr>
<td>∅+ 1</td>
<td>{a}+ .7</td>
<td>{a, c}+ .4</td>
<td>{a, c, d}+** .3</td>
</tr>
<tr>
<td></td>
<td>{b} .3</td>
<td>{a, d}+ .5</td>
<td>{a, c, e}+** .3</td>
</tr>
<tr>
<td></td>
<td>{c}+ .7</td>
<td>{a, e}+ .6</td>
<td>{a, d, e}+** .4</td>
</tr>
<tr>
<td></td>
<td>{d}+ .6</td>
<td>{b, c}+** .3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{e}+ .7</td>
<td>{c, d}+ .4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{c, e}+ .4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{d, e} .4</td>
<td></td>
</tr>
</tbody>
</table>

- **Frequent Item Set:** any frequent item set (support is higher than the minimal support)
- **Closed Item Set:** (marked with +) a frequent item set is called *closed* if no superset has the same support
- **Maximal Item Set:** (marked with *) a frequent item set is called *maximal* if no superset is frequent
Searching for Frequent Item Sets

- it suffices to find closed item sets together with their support
- characterization of closed item sets by

\[ I \text{ closed } \iff s_T(I) \geq s_{\text{min}} \land I = \bigcap_{k \in K_T(I)} t_k \]

suggests to find them by forming all possible intersections of the transactions and checking their support
- however, approaches using this idea are not competitive with other methods
- if support of all frequent item sets is needed, it can be clumsy and tedious to compute support of non-closed frequent item set with

\[ \forall s_{\text{min}} : \forall I \in F_T(s_{\text{min}}) - C_T(s_{\text{min}}) : s_T(I) = \max_{J \in C_T(s_{\text{min}}), J \supset I} s_T(J) \]

- in order to find closed sets one may have to visit many frequent sets anyway
Finding the Frequent Item Sets

**idea:** use properties of support to organize search for all frequent item sets

\[
\forall I : \forall J \supset I : \\
s_T(I) < s_{\text{min}} \\
\rightarrow s_T(J) < s_{\text{min}}
\]

since these properties relate support of item set to support of its **subsets** and **supersets**, organize search based on **subset lattice** of set \( A \) (set of all items)

Hasse diagram
Subset Lattice and Frequent Item Sets

transaction vector

1: \{a, d, e\}
2: \{b, c, d\}
3: \{a, c, e\}
4: \{a, c, d, e\}
5: \{a, e\}
6: \{a, c, d\}
7: \{b, c\}
8: \{a, c, d, e\}
9: \{c, b, e\}
10: \{a, d, e\}

blue boxes are frequent item sets, white boxes are infrequent item sets

subset lattice with frequent item sets \((s_{\text{min}} = 3)\):
Subset Lattice and Closed Item Sets

transaction vector

1: \{a, d, e\}
2: \{b, c, d\}
3: \{a, c, e\}
4: \{a, c, d, e\}
5: \{a, e\}
6: \{a, c, d\}
7: \{b, c\}
8: \{a, c, d, e\}
9: \{c, b, e\}
10: \{a, d, e\}

red boxes are closed item sets, white boxes are infrequent item sets

subset lattice with closed item sets

\(s_{\text{min}} = 3\):
Subset Lattice and Maximal Item Sets

transaction vector

1: \{a, d, e\}
2: \{b, c, d\}
3: \{a, c, e\}
4: \{a, c, d, e\}
5: \{a, e\}
6: \{a, c, d\}
7: \{b, c\}
8: \{a, c, d, e\}
9: \{c, b, e\}
10: \{a, d, e\}

red boxes are maximal item sets, white boxes are infrequent item sets

subset lattice with maximal item sets ($s_{\text{min}} = 3$):
Searching for Frequent Item Sets
[Agrawal and Srikant, 1994]

one possible scheme for search:

• determine support of one element item sets and discard infrequent items
• form candidate item sets with two items (both items must be frequent), determine their support, discard infrequent item sets
• form candidate item sets with three items (all pairs must be frequent), determine their support, discard infrequent item sets
• continue by forming candidate item sets with four, five etc. items until no candidate item set is frequent

this is the **Apriori Algorithm** which is based on two main steps: **candidate generation** and **pruning**

all frequent item set mining algorithms are based on these steps in some form
function apriori (A, T, s_{\text{min}}) 
begin 
\hspace{10pt} k := 1; 
\hspace{10pt} E_k := \bigcup_{a \in A}\{\{a\}\}; 
\hspace{10pt} F_k := \text{prune}(E_k, T, s_{\text{min}}); 
while F_k \neq \emptyset do begin 
\hspace{20pt} E_{k+1} := \text{candidates}(F_k); 
\hspace{20pt} F_{k+1} := \text{prune}(E_{k+1}, T, s_{\text{min}}); 
\hspace{20pt} k := k + 1; 
end; 
return \bigcup_{j=1}^{k} F_j; 
end (* apriori *)
(* Apriori algorithm *)
(* initialize the item set size *)
(* start with single element sets *)
(* and determine the frequent ones *)
(* while there are frequent item sets *)
(* create item sets with one item more *)
(* and determine the frequent ones *)
(* increment the item counter *)
(* return the frequent item sets *)
The Apriori Algorithm 2

function candidates \((F_k)\) (* generate candidates with \(k + 1\) items *)
begin
\[ E := \emptyset; \] (* initialize the set of candidates *)
forall \(f_1, f_2 \in F_k\) (* traverse all pairs of frequent item sets *)
with \(f_1 = \{a_1, \ldots, a_{k-1}, a_k\}\) (* that differ only in one item and *)
and \(f_2 = \{a_1, \ldots, a_{k-1}, a'_k\}\) (* are in a lexicographic order *)
and \(a_k < a'_k\) do begin (* (the order is arbitrary, but fixed) *)
\[ f := f_1 \cup f_2 = \{a_1, \ldots, a_{k-1}, a_k, a'_k\}; \] (* union has \(k + 1\) items *)
if \(\forall a \in f : f - \{a\} \in F_k\) (* only if all subsets are frequent, *)
then \(E := E \cup \{f\};\) (* add the new item set to the candidates *)
end;
return \(E;\) (* otherwise it cannot be frequent *)
end (* candidates *)
The Apriori Algorithm 3

function prune \((E, T, s_{\text{min}})\)

begin

forall \(e \in E\) do

\(s_T(e) := 0;\)

forall \(t \in T\) do

forall \(e \in E\) do

if \(e \subseteq t\)

then \(s_T(e) := s_T(e) + 1;\)

\(F := \emptyset;\)

forall \(e \in E\) do

if \(s_T(e) \geq s_{\text{min}}\)

then \(F := F \cup \{e\};\)

return \(F;\)

end (* prune *)

(* prune infrequent candidates *)

(* initialize the support counters *)

(* of all candidates to be checked *)

(* traverse the transactions *)

(* traverse the candidates *)

(* if transaction contains the candidate, *)

(* increment the support counter *)

(* initialize the set of frequent candidates *)

(* traverse the candidates *)

(* if a candidate is frequent, *)

(* add it to the set of frequent candidates *)

(* return the pruned set of candidates *)
Searching for Frequent Item Sets

- Apriori algorithm searches subset lattice top-down level by level.
- Collecting frequent item sets of size $k$ in set $F_k$ has drawbacks: frequent item set of size $k + 1$ can be formed in 
  \[ j = \frac{k(k + 1)}{2} \]
  possible ways (for infrequent item sets, number may be smaller).
- Consequence: candidate generation step may carry out a lot of redundant work, since it suffices to generate each candidate item set once.
- **Question:** can we reduce or even eliminate this redundant work?
- **More generally:** how can we make sure that any candidate item set is generated at most once?
- **Idea:** assign to each item set unique parent item set, from which this item set is to be generated.
Searching for Frequent Item Sets

• core problem: item set of size $k$ (i.e. with $k$ items) can be generated in $k!$ different ways (on $k!$ paths in Hasse diagram), because in principle items may be added in any order
• if we consider an item by item process of building item set (levelwise traversal of the lattice), there are $k$ possible ways of forming an item set of size $k$ from item sets of size $k - 1$ by adding the remaining item
• obvious: it suffices to consider each item set at most once in order to find frequent ones (infrequent item sets need not be generated at all)
• question: can we reduce or even eliminate this variety? more generally: how can we make sure that any candidate item set is generated at most once?
• idea: assign to each item set a unique parent item set, from which this item set is to be generated
Searching for Frequent Item Sets

- we must search item subset lattice / its Hasse diagram
- assigning unique parents turns Hasse diagram into tree
- traversing resulting tree explores each item set exactly once

subset lattice (Hasse diagram) and possible tree for five items:
Searching with Unique Parents

Principle of Search Algorithm based on Unique Parents:

- **Base Loop:**
  - traverse all one-element item sets (their unique parent is $\emptyset$)
  - recursively process all one-element item sets that are frequent

- **Recursive Processing:**
  for given frequent item set $I$:
  - generate all extensions $J$ of $I$ by one item (i.e. $J \supset I$, $|J| = |I| + 1$)
    for which item set $I$ is chosen unique parent
  - for all $J$: if $J$ is frequent, process $J$ recursively, otherwise discard $J$

- **Questions:**
  - how can we formally assign unique parents?
  - how can we make sure that we generate only those extensions for which item set that is extended is chosen unique parent?
Unique Parents and Prefix Trees

- item sets sharing same longest proper prefix are siblings, because they have same unique parent
- this allows to represent unique parent tree as **prefix tree** or trie

canonical parent tree and corresponding prefix tree for 5 items:
Apriori: Levelwise Search

1: \{a, d, e\}
2: \{b, c, d\}
3: \{a, c, e\}
4: \{a, c, d, e\}
5: \{a, e\}
6: \{a, c, d\}
7: \{b, c\}
8: \{a, c, d, e\}
9: \{c, b, e\}
10: \{a, d, e\}

- example transaction database with 5 items and 10 transactions
- minimum support: 30\%, \textit{i.e.} at least 3 transactions must contain item set
- all one item sets are frequent \(\rightarrow\) full second level is needed
Apriori: Levelwise Search

1: \{a, d, e\}
2: \{b, c, d\}
3: \{a, c, e\}
4: \{a, c, d, e\}
5: \{a, e\}
6: \{a, c, d\}
7: \{b, c\}
8: \{a, c, d, e\}
9: \{c, b, e\}
10: \{a, d, e\}

- determining support of item sets: for each item set traverse database and count transactions that contain it (highly inefficient)
- better: traverse tree for each transaction and find item sets it contains (efficient: can be implemented as simple doubly recursive procedure)
Apriori: Levelwise Search

1: \{a, d, e\}
2: \{b, c, d\}
3: \{a, c, e\}
4: \{a, c, d, e\}
5: \{a, e\}
6: \{a, c, d\}
7: \{b, c\}
8: \{a, c, d, e\}
9: \{c, b, e\}
10: \{a, d, e\}

- minimum support: 30%, i.e. at least 3 transactions must contain item set
- infrequent item sets: \{a, b\}, \{b, d\}, \{b, e\}
- subtrees starting at these item sets can be pruned
Apriori: Levelwise Search

1: \{a, d, e\}
2: \{b, c, d\}
3: \{a, c, e\}
4: \{a, c, d, e\}
5: \{a, e\}
6: \{a, c, d\}
7: \{b, c\}
8: \{a, c, d, e\}
9: \{c, b, e\}
10: \{a, d, e\}

• generate candidate item sets with 3 items (parents must be frequent)
• before counting, check whether candidates contain infrequent item set
  • item set with \(k\) items has \(k\) subsets of size \(k - 1\)
  • parent is only one of these subsets
Apriori: Levelwise Search

1: \{a, d, e\}
2: \{b, c, d\}
3: \{a, c, e\}
4: \{a, c, d, e\}
5: \{a, e\}
6: \{a, c, d\}
7: \{b, c\}
8: \{a, c, d, e\}
9: \{c, b, e\}
10: \{a, d, e\}

• item sets \{b, c, d\} and \{b, c, e\} can be pruned, because
  • \{b, c, d\} contains infrequent item set \{b, d\} and
  • \{b, c, e\} contains infrequent item set \{b, e\}
Apriori: Levelwise Search

1: \{a, d, e\}
2: \{b, c, d\}
3: \{a, c, e\}
4: \{a, c, d, e\}
5: \{a, e\}
6: \{a, c, d\}
7: \{b, c\}
8: \{a, c, d, e\}
9: \{c, b, e\}
10: \{a, d, e\}

- only remaining 4 item sets of size 3 are evaluated
Apriori: Levelwise Search

1: \{a, d, e\}
2: \{b, c, d\}
3: \{a, c, e\}
4: \{a, c, d, e\}
5: \{a, e\}
6: \{a, c, d\}
7: \{b, c\}
8: \{a, c, d, e\}
9: \{c, b, e\}
10: \{a, d, e\}

- minimum support: 30\%, i.e. at least 3 transactions must contain item set
- infrequent item set: \{c, d, e\}
Apriori: Levelwise Search

1: \{a, d, e\}
2: \{b, c, d\}
3: \{a, c, e\}
4: \{a, c, d, e\}
5: \{a, e\}
6: \{a, c, d\}
7: \{b, c\}
8: \{a, c, d, e\}
9: \{c, b, e\}
10: \{a, d, e\}

- generate candidate item sets with 4 items (parents must be frequent)
- before counting, check whether candidates contain an infrequent item set
Apriori: Levelwise Search

1: \{a, d, e\}
2: \{b, c, d\}
3: \{a, c, e\}
4: \{a, c, d, e\}
5: \{a, e\}
6: \{a, c, d\}
7: \{b, c\}
8: \{a, c, d, e\}
9: \{c, b, e\}
10: \{a, d, e\}

- item set \{a, c, d, e\} can be pruned, because it contains infrequent item set \{c, d, e\}
- consequence: no candidate item sets with four items
- fourth access to transaction database is not necessary
Summary Apriori

Basic Processing Scheme

- breadth-first/levelwise traversal of the subset lattice
- candidates are formed by merging item sets that differ in only one item
- support counting is done with doubly recursive procedure

Advantages

- “perfect” pruning of infrequent candidate item sets (with infrequent subsets)

Disadvantages

- can require lots of memory (since all frequent item sets are represented)
- support counting takes very long for large transactions

Software

- http://www.borgelt.net/apriori.html
Summary Frequent Item Set Mining

- many different algorithms for frequent item set mining exist
- here only Apriori algorithm

-algorithms for frequent item set mining differ in:
  - **traversal order** of prefix tree: (breadth-first/levelwise vs. depth-first traversal)
  - **transaction representation**: horizontal (item arrays) vs. vertical (transaction lists) vs. specialized data structures like FP-trees
  - **types of frequent item sets** found: frequent vs. closed vs. maximal item sets (additional pruning methods for closed and maximal item sets)

- **additional filtering** is necessary to reduce size of output (not discussed in this talk)
Association Rules: Basic Notions

- often found patterns are expressed as association rules, e.g.
  - If a customer buys bread and wine,
  - then she/he will probably also buy cheese.

- formally, we consider rules of form $X \rightarrow Y$, with $X, Y \subseteq A$ and $X \cap Y = \emptyset$

- support of rule $X \rightarrow Y$:
  - either: $\varsigma_T(X \rightarrow Y) = \sigma_T(X \cup Y)$ (more common: correct rule)
  - or: $\varsigma_T(X \rightarrow Y) = \sigma_T(X)$ (more plausible: applicable rule)

- confidence of rule $X \rightarrow Y$:
  $$ c_T(X \rightarrow Y) = \frac{\sigma_T(X \cup Y)}{\sigma_T(X)} = \frac{s_T(X \cup Y)}{s_T(X)} = \frac{s_T(I)}{s_T(X)} $$
  can be seen as estimate of $P(Y \mid X)$
Association Rules: Formal Definition

given:

- set $A = \{a_1, \ldots, a_m\}$ of items,
- vector $T = (t_1, \ldots, t_n)$ of transactions over $A$,
- real number $\varsigma_{\min}$, $0 < \varsigma_{\min} \leq 1$, minimum support,
- real number $c_{\min}$, $0 < c_{\min} \leq 1$, minimum confidence

desired:

- set of all association rules, i.e. set

\[
\mathcal{R} = \{ R : X \rightarrow Y \mid \varsigma_T(R) \geq \varsigma_{\min} \wedge c_T(R) \geq c_{\min} \}
\]

general procedure:

- find frequent item sets
- construct rules and filter them w.r.t. $\varsigma_{\min}$ and $c_{\min}$
Generating Association Rules

- which minimum support has to be used for finding frequent item sets depends on definition of support of rule:
  - if $\varsigma_T(X \rightarrow Y) = \sigma_T(X \cup Y)$, then $\sigma_{\text{min}} = \varsigma_{\text{min}}$ or equivalently $s_{\text{min}} = \lceil n\varsigma_{\text{min}} \rceil$
  - if $\varsigma_T(X \rightarrow Y) = \sigma_T(X)$, then $\sigma_{\text{min}} = \varsigma_{\text{min}}c_{\text{min}}$ or equivalently $s_{\text{min}} = \lceil n\varsigma_{\text{min}}c_{\text{min}} \rceil$

- after frequent item sets have been found, rule construction then traverses all frequent item sets $I$ and splits them into disjoint subsets $X$ and $Y$ ($X \cap Y = \emptyset$ and $X \cup Y = I$), thus forming rules $X \rightarrow Y$
  - filtering rules w.r.t. confidence is always necessary
  - filtering rules w.r.t. support is only necessary if $\varsigma_T(X \rightarrow Y) = \sigma_T(X)$
Properties of the Confidence

- from \(\forall I : \forall J \subseteq I : s_T(I) \leq s_T(J)\) it obviously follows

\[
\forall X, Y : \forall a \in X : \quad \frac{s_T(X \cup Y)}{s_T(X)} \geq \frac{s_T(X \cup Y)}{s_T(X - \{a\})}
\]

and therefore

\[
\forall X, Y : \forall a \in X : \quad c_T(X \rightarrow Y) \geq c_T(X - \{a\} \rightarrow Y \cup \{a\}),
\]

i.e. moving an item from antecedent to consequent cannot increase confidence of rule

- immediate consequence:

\[
\forall X, Y : \forall a \in X : \quad c_T(X \rightarrow Y) < c_{\min} \rightarrow c_T(X - \{a\} \rightarrow Y \cup \{a\}) < c_{\min}
\]

i.e. if rule fails to meet minimum confidence, then no rules over same item set and with larger consequent need to be considered
**Generating Association Rules**

function rules \( (F) \);

\[ R := \emptyset; \]

forall \( f \in F \) do begin

\[ m := 1; \]

\[ H_m := \bigcup_{i \in f} \{ \{ i \} \}; \]

repeat

forall \( h \in H_m \) do

if \( \frac{s_T(f)}{s_T(f-h)} \geq c_{\text{min}} \) then

\[ R := R \cup \{ [(f - h) \rightarrow h] \}; \]  

else \( H_m := H_m - \{ h \}; \)

\( H_{m+1} := \text{candidates}(H_m); \)

\[ m := m + 1; \]

until \( H_m = \emptyset \) or \( m \geq |f| \);

end;

return \( R \); end; (* rules *)
Generating Association Rules

function candidates \((F_k)\) \hspace{1cm} (* generate candidates with \(k + 1\) items *)
begin

\(E := \emptyset;\) \hspace{1cm} (* initialize the set of candidates *)

forall \(f_1, f_2 \in F_k\) \hspace{1cm} (* traverse all pairs of frequent item sets *)
with \(f_1 = \{a_1, \ldots, a_{k-1}, a_k\}\) \hspace{1cm} (* that differ only in one item and *)
and \(f_2 = \{a_1, \ldots, a_{k-1}, a'_k\}\) \hspace{1cm} (* are in a lexicographic order *)
and \(a_k < a'_k\) do begin

\(f := f_1 \cup f_2 = \{a_1, \ldots, a_{k-1}, a_k, a'_k\};\) (* union has \(k + 1\) items *)
if \(\forall a \in f: f - \{a\} \in F_k\) (* only if all subsets are frequent, *)
then \(E := E \cup \{f\};\) (* add the new item set to the candidates *)
end;

return \(E;\) (* return the generated candidates *)

end (* candidates *)
Frequent Item Sets: Example

transaction vector

1: \{a, d, e\}
2: \{b, c, d\}
3: \{a, c, e\}
4: \{a, c, d, e\}
5: \{a, e\}
6: \{a, c, d\}
7: \{b, c\}
8: \{a, c, d, e\}
9: \{c, b, e\}
10: \{a, d, e\}

frequent item sets

<table>
<thead>
<tr>
<th>0 items</th>
<th>1 item</th>
<th>2 items</th>
<th>3 items</th>
</tr>
</thead>
<tbody>
<tr>
<td>\emptyset: 1</td>
<td>{a}: .7</td>
<td>{a, c}: .4</td>
<td>{a, c, d}: .3</td>
</tr>
<tr>
<td></td>
<td>{b}: .3</td>
<td>{a, d}: .5</td>
<td>{a, c, e}: .3</td>
</tr>
<tr>
<td></td>
<td>{c}: .7</td>
<td>{a, e}: .6</td>
<td>{a, d, e}: .4</td>
</tr>
<tr>
<td></td>
<td>{d}: .6</td>
<td>{b, c}: .3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{e}: .7</td>
<td>{c, d}: .4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{c, e}: .4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{d, e}: .4</td>
<td></td>
</tr>
</tbody>
</table>

- minimum support is $s_{\text{min}} = 3$ or $\sigma_{\text{min}} = 0.3 = 30\%$ in this example
- $2^5 = 32$ possible item sets over $A = \{a, b, c, d, e\}$
- 16 frequent item sets (but only 10 transactions)
Generating Association Rules

example: \( I = \{a, c, e\}, \ X = \{c, e\}, \ Y = \{a\} \)

\[
c_T(c, e \rightarrow a) =
\]

minimum confidence: 80%

<table>
<thead>
<tr>
<th>association rule</th>
<th>support of all items</th>
<th>support of antecedent</th>
<th>confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>( b \rightarrow c: )</td>
<td>.3</td>
<td>.3</td>
<td>1</td>
</tr>
<tr>
<td>( d \rightarrow a: )</td>
<td>.5</td>
<td>.6</td>
<td>.833</td>
</tr>
<tr>
<td>( e \rightarrow a: )</td>
<td>.6</td>
<td>.7</td>
<td>.857</td>
</tr>
<tr>
<td>( a \rightarrow e: )</td>
<td>.6</td>
<td>.7</td>
<td>.857</td>
</tr>
<tr>
<td>( d, e \rightarrow a: )</td>
<td>.4</td>
<td>.4</td>
<td>1</td>
</tr>
<tr>
<td>( a, d \rightarrow e: )</td>
<td>.4</td>
<td>.5</td>
<td>.8</td>
</tr>
</tbody>
</table>
Generating Association Rules

example: \( I = \{a, c, e\}, \ X = \{c, e\}, \ Y = \{a\} \)

\[
c_T(c, e \rightarrow a) = \frac{s_T(\{a, c, e\})}{s_T(\{c, e\})} =
\]

minimum confidence: 80%

<table>
<thead>
<tr>
<th>association rule</th>
<th>support of all items</th>
<th>support of antecedent</th>
<th>confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>( b \rightarrow c: )</td>
<td>.3</td>
<td>.3</td>
<td>1</td>
</tr>
<tr>
<td>( d \rightarrow a: )</td>
<td>.5</td>
<td>.6</td>
<td>.833</td>
</tr>
<tr>
<td>( e \rightarrow a: )</td>
<td>.6</td>
<td>.7</td>
<td>.857</td>
</tr>
<tr>
<td>( a \rightarrow e: )</td>
<td>.6</td>
<td>.7</td>
<td>.857</td>
</tr>
<tr>
<td>( d, e \rightarrow a: )</td>
<td>.4</td>
<td>.4</td>
<td>1</td>
</tr>
<tr>
<td>( a, d \rightarrow e: )</td>
<td>.4</td>
<td>.5</td>
<td>.8</td>
</tr>
</tbody>
</table>
Generating Association Rules

example: \( I = \{a, c, e\}, X = \{c, e\}, Y = \{a\} \)

\[
c_T(c, e \rightarrow a) = \frac{s_T(\{a, c, e\})}{s_T(\{c, e\})} = \frac{30\%}{40\%} =
\]

minimum confidence: 80%
Generating Association Rules

example: \( l = \{a, c, e\}, X = \{c, e\}, Y = \{a\} \)

\[
c_T(c, e \rightarrow a) = \frac{s_T(\{a, c, e\})}{s_T(\{c, e\})} = \frac{30\%}{40\%} = 75\%
\]

minimum confidence: 80%

<table>
<thead>
<tr>
<th>association rule</th>
<th>support of all items</th>
<th>support of antecedent</th>
<th>confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>( b \rightarrow c:)</td>
<td>.3</td>
<td>.3</td>
<td>1</td>
</tr>
<tr>
<td>( d \rightarrow a:)</td>
<td>.5</td>
<td>.6</td>
<td>.833</td>
</tr>
<tr>
<td>( e \rightarrow a:)</td>
<td>.6</td>
<td>.7</td>
<td>.857</td>
</tr>
<tr>
<td>( a \rightarrow e:)</td>
<td>.6</td>
<td>.7</td>
<td>.857</td>
</tr>
<tr>
<td>( d, e \rightarrow a:)</td>
<td>.4</td>
<td>.4</td>
<td>1</td>
</tr>
<tr>
<td>( a, d \rightarrow e:)</td>
<td>.4</td>
<td>.5</td>
<td>.8</td>
</tr>
</tbody>
</table>
Support of an Association Rule

The two rule support definitions are not equivalent:

Transaction vector

1: \{a, c, e\}
2: \{b, d\}
3: \{b, c, d\}
4: \{a, e\}
5: \{a, b, c, d\}
6: \{c, e\}
7: \{a, b, d\}
8: \{a, c, d\}

Two association rules

<table>
<thead>
<tr>
<th>Association rule</th>
<th>Support of all items</th>
<th>Support of antecedent</th>
<th>Confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a \rightarrow c)</td>
<td>3 (37.5%)</td>
<td>5 (62.5%)</td>
<td>67.7%</td>
</tr>
<tr>
<td>(b \rightarrow d)</td>
<td>4 (50.0%)</td>
<td>4 (50.0%)</td>
<td>100.0%</td>
</tr>
</tbody>
</table>

Let minimum confidence be \(c_{\text{min}} = 0.65\)

- For \(\varsigma_T(R) = \sigma(X \cup Y)\) and \(3 < c_{\text{min}} \leq 4\) only rule \(b \rightarrow d\) is generated, but not rule \(a \rightarrow c\)
- For \(\varsigma_T(R) = \sigma(X)\) there is no value \(c_{\text{min}}\) that generates only rule \(b \rightarrow d\), but not at same time also rule \(a \rightarrow c\)
Additional Rule Filtering

Simple Measures

- general idea: compare \( \hat{p}_T(Y \mid X) = c_T(X \rightarrow Y) \)
  and \( \hat{p}_T(Y) = c_T(\emptyset \rightarrow Y) = \sigma_T(Y) \)
- (absolute) confidence difference to prior:
  \[ d_T(R) = |c_T(X \rightarrow Y) - \sigma_T(Y)| \]
- (absolute) difference of confidence quotient to 1:
  \[ q_T(R) = \left| 1 - \min \left\{ \frac{c_T(X \rightarrow Y)}{\sigma_T(Y)}, \frac{\sigma_T(Y)}{c_T(X \rightarrow Y)} \right\} \right| \]
- confidence to prior ratio (lift):
  \[ l_T(R) = \frac{c_T(X \rightarrow Y)}{\sigma_T(Y)} \]
Additional Rule Filtering

More Sophisticated Measures

- consider \(2 \times 2\) contingency table or estimated probability table:

<table>
<thead>
<tr>
<th></th>
<th>(X \not\subseteq t)</th>
<th>(X \subseteq t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Y \not\subseteq t)</td>
<td>(n_{00})</td>
<td>(n_{01})</td>
</tr>
<tr>
<td>(Y \subseteq t)</td>
<td>(n_{10})</td>
<td>(n_{11})</td>
</tr>
<tr>
<td>(n_{.0})</td>
<td>(n_{.1})</td>
<td>(n_{..})</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th></th>
<th>(X \not\subseteq t)</th>
<th>(X \subseteq t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Y \not\subseteq t)</td>
<td>(p_{00})</td>
<td>(p_{01})</td>
</tr>
<tr>
<td>(Y \subseteq t)</td>
<td>(p_{10})</td>
<td>(p_{11})</td>
</tr>
<tr>
<td>(p_{.0})</td>
<td>(p_{.1})</td>
<td>1</td>
</tr>
</tbody>
</table>

- \(n_{..}\) total number of transactions
  - \(n_{1.}\) number of transactions to which rule is applicable
  - \(n_{11}\) number of transactions for which rule is correct

  \(i.e.\) \(p_{ij} = \frac{n_{ij}}{n_{..}}, \quad p_{i.} = \frac{n_{i.}}{n_{..}}, \quad p_{.j} = \frac{n_{.j}}{n_{..}}\) for \(i, j = 1, 2\)

- general idea: use measures for strength of dependence of \(X\) and \(Y\)
An Information-theoretic Evaluation Measure

Information Gain [Kullback and Leibler, 1951, Quinlan, 1986]

based on Shannon entropy

\[ H = - \sum_{i=1}^{n} p_i \log_2 p_i \]  

[Shannon, 1948]

\[ I_{\text{gain}}(X, Y) = H(Y) - H(Y|X) \]

\[ = - \sum_{i=1}^{k_Y} p_i \cdot \log_2 p_i \quad - \quad \sum_{j=1}^{k_X} p_j \cdot \left( - \sum_{i=1}^{k_Y} p_{i|j} \log_2 p_{i|j} \right) \]

- \( H(Y) \) entropy of distribution of \( Y \)
- \( H(Y|X) \) expected entropy of distribution of \( Y \) if value of \( X \) becomes known
- \( H(Y) - H(Y|X) \) expected entropy reduction or information gain
Summary Association Rules

• **association rule induction is a two step process**
  - find frequent item sets (minimum support)
  - form relevant association rules (minimum confidence)

• **generating association rules**
  - form all possible association rules from frequent item sets
  - filter “interesting” association rules based on minimum support and minimum confidence

• **filtering association rules**
  - compare rule confidence and consequent support
  - information gain
  - other measures, e.g. $\chi^2$ measure, ...
Industrial Applications

- **car manufacturer** collects servicing tasks on all their vehicles
  - what are interesting subgroups of cars?
  - how do these subgroups behave over time?
  - which cars’ suspension failure rate is strongly increasing in winter?

- **bank** assesses credit contracts *w.r.t.* terminability
  - what changes were there in past?
  - any common factors?
  - how to communicate this to non-statisticians, *e.g.* bankers?

- tracking user activity in **virtual environment**
  - are there any oddities in user behavior?
  - how to parameterize “odd” things?
Or: What they have and what they want

data are

- high-dimensional
- many-valued
- time-stamped

results should be

- easy-to-understand patterns (rules)
- exploratory tools (visualization and inspection)
- natural way of interaction
- exploit temporal information (if desired)
Rule Icons

- every rule

\[ \langle A_1 = a_1 \land \cdots \land A_k = a_k \rangle \rightarrow C = c \]

of given rule set is represented as icon

- for every possible item, reserved segment on outer border
Rule Icons

• every rule

$$\langle A_1 = a_1 \land \cdots \land A_k = a_k \rangle \rightarrow C = c$$

of given rule set is represented as icon

• for every possible item, reserved segment on outer border
• if item is present in antecedent, segment is colored
Rule Icons

- every rule

\[ \langle A_1 = a_1 \land \cdots \land A_k = a_k \rangle \rightarrow C = c \]

of given rule set is represented as icon

- for every possible item, reserved segment on outer border
- if item is present in antecedent, segment is colored
- interior encodes rule-measure: e.g. confidence
Rule Icons: Overlapping

- cover of 2 rules may be non-empty
- percentage bar to display mutual overlap
- special case: inclusion

Gender = male → Cancer = yes
Gender = male ∧ Smoker = yes → Cancer = yes
Rule Icons: Overlapping

- cover of 2 rules may be non-empty
- percentage bar to display mutual overlap
- general case:
Rule Icons: Location

- finally, arrange icons in two-dimensional chart
- choose 3 association rule measures for both axes and size of icon
- our suggestion for rule $X \rightarrow Y$, choose following measures:
  - $x$-coordinate: recall, i.e. $c_T(Y \rightarrow X)$
  - $y$-coordinate: lift, i.e. $c_T(X \rightarrow Y)/\sigma_T(Y)$
  - size: support, i.e. $\sigma_T(X \cup Y)$
Real-world Example: Daimler AG

car database
- 300,000 vehicles
- subset of 180 attributes
- 2–300 values per attribute
- probabilistic dependency network
Real-world Example: Daimler AG

![Graph showing lift vs. recall with three clusters labeled σ₁, σ₂, and σ₃. The x-axis is labeled recall(Temperatur, Laufleistung | Klassenvariable) and the y-axis is labeled lift(Klassenvariable | Temperatur, Laufleistung).]
Real-world Example: Daimler AG

Explorative Analysis Tool
Real-world Example: ADAC

customer database

- car and customer information
- assessment of vehicle quality
Temporal Change of Rules

- **why considering temporal development of rules?**
  - (i.e. change of certain rule evaluation measures)
  - failure patterns usually do not arise out of sudden, but rather evolve slowly over time
  - fixed problem takes some while to have measurable effect

- **how to present this evolution to user?**
  - create time series for every measure used for locating and scaling rule icon
  - interpolate between frames and present animation

- **problem:** high number of rules
How does that look like?

real-world dataset
How does that look like?

obviously, there is demand for post-processing rule set
Temporal Change of Rules

1. divide dataset into reasonable time frames
2. run respective pattern induction algorithm
3. quantify each pattern \( w.r.t. \) any desired measure(s)
4. generate time series for each measure and each pattern
5. **match time series against user-specified concept**
6. rank them according to membership of concept
User-driven Post-processing

- often users have idea in which direction to investigate
- however, they cannot explicitly phrase query for data mining
- we can use “fuzzy” intentions to thin out rule set, e.g.
  
  “Show me only those rules that had a strongly increasing support and an increasing confidence in the last quarter.”

  or

  “Which patterns exhibit an increasing lift while the support was stable or at most slightly decreasing?”
User-driven Post-processing

1. specify fuzzy partition on change rate domain of every pattern evaluation measure
User-driven Post-processing

1. specify fuzzy partition on change rate domain of every pattern evaluation measure
2. encode user-concept as fuzzy antecedent

\[ \langle \Delta_{\text{lift is unch}} \land \Delta_{\text{conf is incr}} \rangle \]

will be evaluated as

\[ \top \left( \mu_{\Delta_{\text{lift}}}^{\text{unch}}(\tilde{a} \rightarrow c), \mu_{\Delta_{\text{conf}}}^{\text{incr}}(\tilde{a} \rightarrow c) \right) \]

where \( \top \) is t-norm that represents fuzzy conjunction

\[ \text{e.g. “lift is unchanged and confidence is increasing”} \]
User-driven Post-processing

1. specify fuzzy partition on change rate domain of every pattern evaluation measure
2. encode user-concept as fuzzy antecedent
3. order patterns w.r.t. concept membership degrees
Summary Industrial Applications

requirements

- easy-to-understand patterns
- exploratory visual tools
- natural and intuitive interaction
- exploitation of temporal information

desired properties of rules

- almost parameter-free (support and confidence have clear notion and can even be increased after induction)
- no black-box approach
- intuitive type of patterns (decision/business rules)
- natural way of treating missing values
- small data preprocessing overhead
Outline

1. Introduction

2. Association Rules and Frequent Item Sets

3. Frequent Sequence Mining
   Canonical Form for Undirected Sequences
   Allen’s Interval Relations
   Temporal Interval Patterns
   Quality Monitoring of Vehicles

4. Finding Motifs in Time Series Effectively
Frequent Sequence Mining

- **directed vs. undirected sequences**
  - e.g. temporal sequences are always directed
  - DNA sequences can be undirected (both directions can be relevant)

- **multiple sequences vs. single sequence**
  - multiple sequences: purchases with rebate cards, web server access protocols
  - single sequence: alarms in telecommunication networks

- **(time) points vs. time intervals**
  - points: DNA sequences, alarms in telecommunication networks
  - intervals: weather data, movement analysis (sports medicine)
  - further distinction: one object per (time) point vs. multiple objects
Frequent Sequence Mining

- consecutive subsequences vs. subsequences with gaps
  - $a\ c\ b\ a\ b\ c\ b\ a$ always counts as subsequence $abc$
  - $a\ c\ b\ a\ b\ c\ b\ c$ may not always count as subsequence $abc$

- existence of occurrence vs. counting occurrences
  - combinatorial counting (all occurrences)
  - maximal number of disjoint occurrences
  - temporal support (number of time window positions)
  - minimum occurrence (smallest interval)

- relation between objects in sequence
  - items: only precede and succeed
  - labeled time points: $t_1 < t_2$, $t_1 = t_2$, and $t_1 > t_2$
  - labeled time intervals: relations like before, starts, overlaps etc.
Frequent Sequence Mining

- **directed sequences** are easier to handle:
  - (sub)sequence itself can be used as code word
  - only 1 possible code word per sequence (only 1 direction) → this code word is necessarily canonical

- **consecutive subsequences** are easier to handle:
  - fewer occurrences of given subsequence
  - for each occurrence, exactly one possible extensions
  - allows specialized data structures (similar to tree)

- **item sequences** are easiest to handle:
  - only 2 possible relations and thus patterns are simple
  - other sequences are handled with state machines for containment tests
A Canonical Form for Undirected Sequences

• if sequences to mine are not directed, subsequence can not be used as its own code word, because it does not have prefix property

• reason: undirected sequence can be read forward or backward

→ two possible code words, smaller (or larger) of which may then be defined as canonical code word

• examples (that prefix property is violated):
  • assume: item order \( a < b < c \ldots \) and lexicographically smaller code word is canonical one
  • sequence \( bab \), which is canonical, has prefix \( ba \), but canonical form of sequence \( ba \) is rather \( ab \)
  • sequence \( cabd \), which is canonical, has the prefix \( cab \), but canonical form of sequence \( cab \) is rather \( bac \)
A Canonical Form for Undirected Sequences

one possibility to form them having prefix property:

- handle (sub)sequences of **even and odd length separately**

- in addition, forming the code word is started **in the middle**

  **even length:** sequence  \( a_m a_{m-1} \ldots a_2 a_1 b_1 b_2 \ldots b_{m-1} b_m \)

  is described by code word  \( a_1 b_1 a_2 b_2 \ldots a_{m-1} b_{m-1} a_m b_m \)

  or by code word  \( b_1 a_1 b_2 a_2 \ldots b_{m-1} a_{m-1} b_m a_m \).

  **odd length:** sequence  \( a_m a_{m-1} \ldots a_2 a_1 a_0 b_1 b_2 \ldots b_{m-1} b_m \)

  is described by code word  \( a_0 a_1 b_1 a_2 b_2 \ldots a_{m-1} b_{m-1} a_m b_m \)

  or by code word  \( a_0 b_1 a_1 b_2 a_2 \ldots b_{m-1} a_{m-1} b_m a_m \).

- the lexicographically smaller of 2 code words is **canonical code word**

- such sequences are **extended** by adding pair  \( a_{m+1} b_{m+1} \) or  \( b_{m+1} a_{m+1} \), i.e. by adding 1 item at front and 1 item at end
A Canonical Form for Undirected Sequences

code words defined in this way have **prefix property**:

- suppose prefix property would *not* hold
- then *w.l.o.g.*, there exists a canonical code word

\[ w_m = a_1 b_1 a_2 b_2 \ldots a_{m-1} b_{m-1} a_m b_m, \]

the prefix \( w_{m-1} \) of which is not canonical, where

\[ w_{m-1} = a_1 b_1 a_2 b_2 \ldots a_{m-1} b_{m-1}, \]

- consequence: \( w_m < v_m \), where

\[ v_m = b_1 a_1 b_2 a_2 \ldots b_{m-1} a_{m-1} b_m a_m, \]

and \( v_{m-1} < w_{m-1} \), where

\[ v_{m-1} = b_1 a_1 b_2 a_2 \ldots b_{m-1} a_{m-1} \]

- but: \( v_{m-1} < w_{m-1} \) implies \( v_m < w_m \), because \( v_{m-1} \) is prefix of \( v_m \) and \( w_{m-1} \) is a prefix of \( w_m \), but \( v_m < w_m \) contradicts \( w_m < v_m \)
A Canonical Form for Undirected Sequences

- generating and comparing 2 possible code words takes linear time
- however, this can be improved by maintaining additional piece of information
- for each sequence, symmetry flag is computed:
  \[ s_m = \bigwedge_{i=1}^{m} (a_i = b_i) \]
- symmetry flag can be maintained in constant time with
  \[ s_{m+1} = s_m \land (a_{m+1} = b_{m+1}) \]
- permissible extensions depend on symmetry flag:
  - if \( s_m = \text{true} \), it must be \( a_{m+1} \leq b_{m+1} \)
  - if \( s_m = \text{false} \), any relation between \( a_{m+1} \) and \( b_{m+1} \) is acceptable
- rule guarantees: exactly canonical extensions are created
- applying this rule to check candidate extension takes constant time
Sequences of Time Intervals

- (labeled or attributed) **time interval** is triple \( l = (s, e, l) \), where \( s \) is start time, \( e \) is end time and \( l \) is associated label.
- **time interval sequence** is set of (labeled) time intervals, of which we assume that they are maximal in sense that for 2 intervals \( l_1 = (s_1, e_1, l_1) \) and \( l_2 = (s_2, e_2, l_2) \) with \( l_1 = l_2 \) we have either \( e_1 < s_2 \) or \( e_2 < s_1 \) (otherwise they are merged into 1 interval \( l = (\min\{s_1, s_2\}, \max\{e_1, e_2\}, l_1) \)).
- **time interval sequence database** is vector of time interval sequences.
- Time intervals can easily be ordered as follows: let \( l_1 = (s_1, e_1, l_1) \) and \( l_2 = (s_2, e_2, l_2) \) be 2 time intervals, it is \( l_1 < l_2 \) iff
  - \( s_1 < s_2 \) or
  - \( s_1 = s_2 \) and \( e_1 < e_2 \) or
  - \( s_1 = s_2 \) and \( e_1 = e_2 \) and \( l_1 < l_2 \)
  due to assumption made above, at least 3rd option must hold.
Allen’s Interval Relations

- due to their temporal extension, time intervals allow for different relations
- commonly used set of relations between time intervals are Allen’s interval relations [Allen, 1983]

- A before B
- A meets B
- A overlaps B
- A is finished by B
- A contains B
- A is started by B
- A equals B
Temporal Interval Patterns

[Kempe et al., 2008]

- pattern must specify relations between all referenced intervals
- this can conveniently be done with matrix:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>e</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>B</td>
<td>?</td>
<td>e</td>
<td>?</td>
</tr>
<tr>
<td>C</td>
<td>?</td>
<td>?</td>
<td>e</td>
</tr>
</tbody>
</table>

- such temporal pattern matrix can also be interpreted as adjacency matrix of graph, which has interval relationships as edge labels
Temporal Interval Patterns
[Kempe et al., 2008]

- pattern must specify relations between all referenced intervals
- this can conveniently be done with matrix:

$$
\begin{array}{ccc}
A & B & C \\
A & e & o & b \\
B & i & o & e & m \\
C & a & i & m & e
\end{array}
$$

- such temporal pattern matrix can also be interpreted as adjacency matrix of graph, which has interval relationships as edge labels
Temporal Interval Patterns
[Kempe et al., 2008]

- pattern must specify relations between all referenced intervals
- this can conveniently be done with matrix:

\[
\begin{array}{c|ccc}
 & A & B & C \\
\hline
A & e & b \\
B & e & m \\
C & im & e \\
\end{array}
\]

- such temporal pattern matrix can also be interpreted as adjacency matrix of graph, which has interval relationships as edge labels
- generally, input interval sequences may be represented as such graphs, thus mapping problem to frequent (sub)graph mining
- however, relationships between time intervals are constrained (e.g. “B after A” and “C after B” imply “C after A”)
- constraints can be exploited to obtain simpler canonical form
- in **canonical form**, intervals are assigned in increasing time order to rows and columns of temporal pattern matrix
Support of Temporal Patterns

- support of temporal pattern \textit{w.r.t.} single sequence can be defined by
  - combinatorial counting (all occurrences)
  - maximal number of disjoint occurrences
  - temporal support (number of time window positions)
  - minimum occurrence (smallest interval)

- however, all of these definitions suffer from fact that such support is not \textit{anti-monotone} or \textit{downward closed}:

\begin{center}
\begin{tabular}{|c|c|}
\hline
B & A \\
\hline
B & B \\
\hline
\end{tabular}
\end{center}
Support of Temporal Patterns

- support of temporal pattern \( w.r.t. \) single sequence can be defined by
  - combinatorial counting (all occurrences)
  - maximal number of disjoint occurrences
  - temporal support (number of time window positions)
  - minimum occurrence (smallest interval)

- however, all of these definitions suffer from fact that such support is not \textit{anti-monotone} or \textit{downward closed}:

  \[
  \begin{array}{ccc}
  B & A & B \\
  \end{array}
  \]

  support of “\( A \) contains \( B \)” is 2, but support of “\( A \)” is only 1
Support of Temporal Patterns

- support of temporal pattern w.r.t. single sequence can be defined by:
  - combinatorial counting (all occurrences)
  - maximal number of disjoint occurrences
  - temporal support (number of time window positions)
  - minimum occurrence (smallest interval)

- however, all of these definitions suffer from fact that such support is not anti-monotone or downward closed:

  ![Example Diagram](image)

  support of “A contains B” is 2, but support of “A” is only 1

- nevertheless, exhaustive pattern search can ensured, without having to abandon pruning with [Apriori property](#)

- reasons: with minimum occurrence counting, relationship “contains” is the only one that can lead to support anomalies
Weakly Anti-Monotone / Downward Closed

[Kempe et al., 2008]

- let $\mathcal{P}$ pattern space with subpattern relationship $\sqsubseteq$ and let $s$ be function from $\mathcal{P}$ to real numbers, $s : \mathcal{P} \rightarrow \mathbb{R}$
- for pattern $S \in \mathcal{P}$, let $P(S) = \{ R \mid R \sqsubseteq S \land \neg \exists Q : R \sqsubseteq Q \sqsubseteq S \}$ be set of all parent patterns of $S$
- function $s$ on pattern space $\mathcal{P}$ is called
  - strongly anti-monotone or strongly downward closed iff
    $\forall S \in \mathcal{P} : \forall R \in P(S) : s(R) \geq s(S)$
  - weakly anti-monotone or weakly downward closed iff
    $\forall S \in \mathcal{P} : \exists R \in P(S) : s(R) \geq s(S)$
- support of temporal interval patterns is weakly anti-monotone (at least) if it is computed from minimal occurrences
- if temporal interval patterns are extended backwards in time, then Apriori property can safely be used for pruning
Summary Frequent Sequence Mining

- different types of frequent sequence mining can be distinguished:
  - single and multiple sequences, directed and undirected sequences
  - items vs. (labeled) intervals, single and multiple objects per position
  - relations between objects, definition of pattern support

- all common types of frequent sequence mining possess canonical forms for which canonical extension rules can be found

- with these rules it is possible to check in constant time whether possible extension leads to result in canonical form

- weakly anti-monotone support function can be enough to allow pruning with Apriori property

- however, in this case: make sure that canonical form assigns appropriate parent pattern to ensure exhaustive search
Quality Monitoring of Vehicles

101,250 vehicles

- garage stops
- vehicle configuration
- 1.4 million temporal intervals

Diesel, Automatik, AMG, ...

SSL 1 am 1.4.2006, SSL 2 am 11.11. 2006, ...

Standheizung

Automatik

SSL 1  SSL 2  SSL 3
Quality Monitoring of Vehicles

- Standheizung
  - SSL 1
  - SSL 2
  - SSL 3

- Automatik

- Kombi
  - SSL 3

- Automatik
  - SSL 1

- Diesel
  - Automatik
  - SSL 1
  - SSL 3
Quality Monitoring of Vehicles

Standheizung

Automatik

SSL 1  SSL 2  SSL 3

Kombi

SSL 3

Automatik

SSL 1

Diesel

Automatik

SSL 1  SSL 3

Confidence: 0.66
Pre-Production Vehicles

**FIRST GEAR**

**REVERSE**

**NEUTRAL**

**MAP sensor voltage**

| m | h | m | h | m | n |
Pre-Production Vehicles
Outline

1. Introduction

2. Association Rules and Frequent Item Sets

3. Frequent Sequence Mining

4. Finding Motifs in Time Series Effectively
   - Time Series Representations
   - Symbolic Aggregate Approximation (SAX)
   - Motifs in Time Series
   - Sub-dimensional Motif: Example
Refresh: Data Mining in Time Series

- Big challenge: to find useful information in time series
- Typical problems: clustering, classification, frequent pattern mining, association rules, visualization, anomaly detection
- Because of huge amount of data, often problems boil down to search for reoccurring similar subsequences
- Needed: similarity measure to compare subsequences
- E.g. Euclidean distance

\[
d(Q, C) = \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2}
\]

of 2 standard normal distributed subsequences \( Q = (q_1, \ldots, q_n)^T \) and \( C = (c_1, \ldots, c_n)^T \)

- Problem: many comparisons, capacity of fast main memory usually too small to load all data
Memory-efficient Representations
[Lin et al., 2007]

- problem: many, slow accesses to raw data
- solution: approximation of time series that fits into main memory and contains interesting features
  - *e.g.* discrete Fourier transformation (DFT), discrete wavelet transformation (DWT), piecewise linear (PLA) or adaptive piecewise constant approximation (APCA), singular value decomposition (SVD)
- here: symbolic representations
- advantage: algorithms from text processing and bioinformatics applicable, *e.g.* hashing, Markov models, suffix trees etc.
Time Series Representations
[Lin et al., 2007]
The most common representations
[Lin et al., 2007]
Piecewise Aggregate Approximation (PAA)  
[Lin et al., 2007]

reduction from 128 to 8 data points
Symbolic Aggregate Approximation (SAX)  
[Lin et al., 2007]

- every sequence of length $n$ becomes a word of defined length $w$ over chosen alphabet $A = \{\alpha_1, \ldots, \alpha_a\}$ with $|A| = a$
- simple algorithm:
  1. separate subsequence into $w$ equally sized intervals
  2. PAA: compute mean value of each interval (as representative) $C = (c_1, \ldots, c_n)^T$ is mapped onto $\bar{C} = (\bar{c}_1, \ldots, \bar{c}_w)$ with

\[
\bar{c}_i = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} c_j
\]

3. map each mean value $\bar{c}_i$ of $\bar{C}$ onto one of $a$ letters with $\hat{a}_i = \alpha_j \iff \beta_{j-1} \leq \bar{c}_i \leq \beta_j$

- assumptions: normally distributed value range of PAA sequence and equiprobable occurrence of each letter
- mapping $\bar{c}_i \mapsto b \in A$ by “cutpoints” $\beta_1, \ldots, \beta_{a-1}$
“Cutpoints” of the Normal Distribution
[Lin et al., 2007]

| \( |A| \) | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|
| \( \beta_1 \) | −.43 | −.67 | −.84 | −.97 | −1.07 | −1.15 | −1.22 | −1.28 |
| \( \beta_2 \) | 0.43 | 0    | 0.25 | 0.43 | 0.57 | 0.67 | 0.76 | 0.84 |
| \( \beta_3 \) | 0.67 | 0.25 | 0    | −.18 | −.32 | −.43 | −.52 |      |
| \( \beta_4 \) | 0.84 | 0.43 | 0.18 | 0    | −.14 | −.25 |      |      |
| \( \beta_5 \) | 0.97 | 0.57 | 0.32 | 0.14 | 0    |      |      |      |
| \( \beta_6 \) | 1.07 | 0.67 | 0.43 | 0.25 |      |      |      |      |
| \( \beta_7 \) | 1.15 | 0.76 | 0.52 |      |      |      |      |      |
| \( \beta_8 \) | 1.22 | 0.84 |      |      |      |      |      |      |
| \( \beta_9 \) |      |      |      |      |      |      |      | 1.28 |

- cutpoints separate normal distribution in equiprobable regions
SAX: Example
[Lin et al., 2007]

- here: $n = 128$, $w = 8$, $a = 3$
- result: \texttt{baabccbc}
SAX: Distance Measure
[Lin et al., 2007]

- **PAA**: lower bound of Euclidean distance with

\[
d_{r}(\bar{Q}, \bar{C}) = \sqrt{\frac{n}{w}} \sqrt{\sum_{i=1}^{w} (\bar{q}_i - \bar{c}_i)^2}
\]

- **SAX**: 

\[
d^{*}(\hat{Q}, \hat{C}) = \sqrt{\frac{n}{w}} \sqrt{\sum_{i=1}^{w} d^*_a(\hat{q}_i, \hat{c}_i)^2}
\]

- **distance** \(d^*_a\) **should** be defined via lookup table, e.g. for \(a = 4\)

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>0.67</td>
<td>1.34</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.67</td>
</tr>
<tr>
<td>c</td>
<td>0.67</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>1.34</td>
<td>0.67</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Comparison of Distances
[Lin et al., 2007]

\[ C \quad Q \]

(A)

\[ \bar{C} \quad \bar{Q} \]

(B)

\[ \hat{C} = b \ a \ a \ b \ c \ c \ b \ c \]
\[ \hat{Q} = b \ a \ b \ c \ a \ c \ c \ a \]  

(C)
SAX Advantage: Lower Bound
[Lin et al., 2007]

- $d^* (\hat{Q}, \hat{C})$ is **lower bound** of Euclidean distance $d(Q, C)$ of original sequences $Q$ and $C$

\[
d^* (\hat{Q}, \hat{C}) \leq d(Q, C)
\]

- if $\hat{Q}$ and $\hat{C}$ are dissimilar, so are $Q$ and $C$
- SAX-based algorithms produce identical result compared to algorithms running on original data
- “only” similar SAX words should be compared in the original space
- usually, only few accesses to original data
Find Motifs in Time Series
[Chiu et al., 2003]

- motifs: primitive, frequent (similar) patterns, prototypes
- challenges:
  - motifs are unknown beforehand
  - complete search is expensive, i.e. $O(n^2)$
  - outliers influence Euclidean distance
Generation of SAX Matrix
[Chiu et al., 2003]

- find all time series motifs of length \( m \) using “sliding window”
- window width \( n \) leads to \((m - n + 1)\) subsequences
- transform every sequence into SAX word of length \( w \)
- save it in row matrix, i.e. SAX matrix
- \( w \) columns, \((m - n + 1)\) rows
Random Projection
[Chiu et al., 2003]

- guess motif positions by so-called random projection
- pairwise comparison of SAX words
- collision matrix $M$ with $(m - n + 1)^2$ cells for all comparisons
- use hash table to implement $M$ efficiently!
- initially, $M(i, j) = 0$ for $1 \leq i, j \leq m - n + 1$
- idea: compare character after character of 2 words in SAX matrix
- assumption: “don’t care symbols” in sequences with unknown position
- e.g. noisy motifs, dilated or contracted sequence
Random Projection
[Chiu et al., 2003]

- thus SAX Matrix is projected onto $1 \leq k < w$ randomly chosen columns
- compare all rows of projected matrix
- if 2 projected SAX words in rows $i$ and $j$ are equal, then increment $M(i, j)$
- repeat projection $t$ times, because it is likely that some motifs will share one cell in $M$ after $t$ iterations
- many random sequences will most likely not collide with already found motif
- user-defined threshold $s$ with $1 \leq s \leq k$ for collision entries in $M$
- all $M(i, j) \geq s$ would be candidate motifs
- but: there are very similar sequences in immediate neighborhood of sequence $i$ (so-called trivial matches)
- these must be removed!
Random Projection: First Two Iterations
[Chiu et al., 2003]
Sub-dimensional Motifs
[Minnen et al., 2007]

• so far: univariate symbolic time series
• random projection can also be used for multivariate symbolic time series
• idea: increment collision matrix $M$ for each variable $j \in \{1, \ldots, p\}$ for each projected SAX word
• problem: relevant dimensions of potential sub-dimensional motifs are unknown
• solution:
  • estimate distribution $P(d_j)$ over distances between non-trivial matches by drawing a sample
  • compute distances $d^*_1, \ldots, d^*_p$ for each entry $M(i,j) \geq s$
  • if $P(d_j \leq d^*_j) < r^\text{rel}_j$ (user-specific dimension relevance), then $j$-th variable will be relevant
Sub-dimensional Motif: Example
[Moewes and Kruse, 2009]

- expert identified $p = 9$ of 130 variables as important
- motifs last at least $n = 400 \text{ ms}$
- given: 10 time series with unknown sub-dimensional motifs
Sub-dimensional Motif in two Time Series
[Moewes and Kruse, 2009]
Clustering of Motifs
[Moewes and Kruse, 2009]

- create dissimilarity matrix by pairwise comparison of all found motifs in 10 time series based on $d^*$
- positive, symmetric matrix with zeros at main diagonal
- can be used to cluster occurrences, which helps finding motifs occurring in several time series
- here: hierarchical clustering of motifs containing variables $\text{attr}_1$ and $\text{attr}_3$
Thanks to...

This talk wouldn’t be possible without the work of

- Steffen Kempe, Daimler AG
- Matthias Steinbrecher, SAP AG

Examples from this talk are based on real-world problems of the following companies:

- ADAC
- Dresdner Bank
- Daimler AG
- Second Life
Literature I

Fast algorithms for mining association rules in large databases.

Maintaining knowledge about temporal intervals.
*Communications of the ACM*, 26:832–843.

Probabilistic discovery of time series motifs.


Zuordnen von linguistischen ausdrücken zu motiven in zeitreihen.

Induction of decision trees.

A mathematical theory of communication.