Data Cube

1. Einführung
2. Aggregation in SQL, GROUP BY
3. Probleme mit GROUP BY
4. Der Cube-Operator
5. Implementierung des Data Cube
6. Zusammenfassung und Ausblick

Dank an Hanna Köpcke!

On-line Analytical Processing (OLAP)

Ziel: Auffinden interessanter Muster in großen Datenmengen

- Formulierung einer Anfrage
- Extraktion der Daten
- Visualisierung der Ergebnisse
- Analyse der Ergebnisse und Formulierung einer neuen Anfrage

OLAP-Werkzeuge

- Datenmenge wird als n-dimensionaler Raum aufgefasst
- Identifizierung von „interessanten" Unterräumen
- In relationalen Datenbanken werden n-dimensionale Daten als Relationen mit n-Attributen modelliert
- Dimensionsreduktion durch Aggregation der Daten entlang der weggelassenen Dimensionen

Beispiel: Autoverkäufe

Modell	Jahr	Farbe	Anzahl
Opel	1990	rot	5
Opel	1990	weiß	87
Opel	1990	blau	62
Opel	1991	rot	54
Opel	1991	weiß	95
Opel	1991	blau	49
Opel	1992	rot	31
Opel	1992	weiß	54
Opel	1992	blau	71
Ford	1990	rot	64
Ford	1990	weiß	62
Ford	1990	blau	63
Ford	1991	rot	52
Ford	1991	weiß	9
Ford	1991	blau	55
Ford	1992	rot	27
Ford	1992	weiß	62
Ford	1992	blau	39

Aggregation in SQL

- Aggregatfunktionen:

  COUNT(), SUM(), MIN(), MAX(), AVG()

  Beispiel: SELECT AVG(Anzahl)
  FROM Autoverkäufe
- Aggregation über verschiedene Werte

  Beispiel: SELECT COUNT(DISTINCT Modell)
  FROM Autoverkäufe
- Aggregatfunktionen liefern einen einzelnen Wert
- Aggregation über mehrere Attribute mit GROUP BY

GROUP BY

SELECT Modell, Jahr, **SUM**(Anzahl)
FROM Autoverkäufe
GROUP BY Modell, Jahr

- Die Tabelle wird gemäß den Kombinationen der ausgewählten Attributmenge in Gruppen unterteilt
- Jede Gruppe wird über eine Funktion aggregiert
- Das Resultat ist eine Tabelle mit aggregierten Werten, indiziert durch die ausgewählte Attributmenge

Beispiel: GROUP BY

Modell	Jahr	Farbe	Anzahl
Opel	1990	rot	5
Opel	1990	weiß	87
Opel	1990	blau	62
Opel	1991	rot	54
Opel	1991	weiß	95
Opel	1991	blau	49
Opel	1992	rot	31
Opel	1992	weiß	54
Opel	1992	blau	71
Ford	1990	rot	64
Ford	1990	weiß	62
Ford	1990	blau	63
Ford	1991	rot	52
Ford	1991	weiß	9
Ford	1991	blau	55
Ford	1992	rot	27
Ford	1992	weiß	62
Ford	1992	blau	39

SELECT Modell, Jahr, **SUM**(Anzahl)
FROM Autoverkäufe
GROUP BY Modell, Jahr

Modell	Jahr	Anzahl
Opel	1990	154
Opel	1991	198
Opel	1992	156
Ford	1990	189
Ford	1991	116
Ford	1992	128

Roll Up

Gleiche Anfrage in unterschiedlichen Detailierungsgraden

- Verminderung des Detailierungsgrades = Roll Up
- Erhöhung des Detailierungsgrades = Drill Down

Beispiel: Autoverkäufe

- Roll Up über drei Ebenen
- Daten werden nach Modell, dann nach Jahr,
  dann nach Farbe aggregiert
- die Verkaufszahlen werden zuerst für jedes Modell
  aus jedem Jahr in jeder Farbe aufgelistet,
  dann werden alle Verkaufszahlen des gleichen Modells
  und Jahres aufsummiert und daraus
  die Verkaufszahlen der Modelle berechnet

GROUP BY: Roll Up

Modell	Jahr	Farbe	Anzahl nach Modell, Jahr, Farbe	Anzahl nach Modell, Jahr	Anzahl nach Modell
Opel	1990	rot	5		
		weiß	87		
		blau	62		
				154	
	1991	rot	54		
		weiß	95		
		blau	49		
				198	
	1992	rot	31		
		weiß	54		
		blau	71		
				156	
					508

Probleme mit GROUP BY: Roll Up

- Tabelle ist nicht relational, da man wegen der leeren Felder (Null-Werte) keinen Schlüssel festlegen kann.
- Die Zahl der Spalten wächst mit der Zahl der aggregierten Attribute
- Um das exponentielle Anwachsen der Spaltenanzahl zu vermeiden, wird der ALL-Wert eingeführt.
- Der ALL-Wert repräsentiert die Menge, über die die Aggregation berechnet wird.

Beispiel:

Ein ALL in der Spalte Farbe bedeutet, dass in der Anzahl dieser Zeile die Verkaufszahlen der roten, weißen und blauen Autos zusammengefasst sind.

GROUP BY: Roll Up mit ALL

Modell	Jahr	Farbe	Anzahl
Opel	1990	rot	5
Opel	1990	weiß	87
Opel	1990	blau	62
Opel	1990	ALL	154
Opel	1991	rot	54
Opel	1991	weiß	95
Opel	1991	blau	49
Opel	1991	ALL	198
Opel	1992	rot	31
Opel	1992	weiß	54
Opel	1992	blau	71
Opel	1992	ALL	156
Opel	ALL	ALL	506

Erzeugung der Tabelle mit SQL:

```
SELECT Modell, ALL, ALL, SUM(Anzahl)
  FROM Autoverkäufe
  WHERE Modell = 'Opel'
  GROUP BY Modell
UNION
  SELECT Modell, Jahr, ALL, SUM(Anzahl)
  FROM Autoverkäufe
  WHERE Modell = 'Opel'
  GROUP BY Modell, Jahr
UNION
  SELECT Modell, Jahr, Farbe, SUM(Anzahl)
  FROM Autoverkäufe
  WHERE Modell = 'Opel'
  GROUP BY Modell, Jahr, Farbe
```

Probleme mit GROUP BY: Roll Up

- Beispiel war ein einfaches dreidimensionales Roll Up
- Eine Aggregation über n Dimensionen erfordert n Unions
- Roll Up ist asymmetrisch:
  Verkäufe sind nach Jahr, aber nicht nach Farbe aggregiert

Kreuztabellen

Symmetrische Darstellung mehrdimensionaler Daten und Aggregationen

Opel	1990	1991	1992	Total (ALL)
rot	5	54	31	90
weiß	87	95	54	236
blau	62	49	71	182
Total (ALL)	154	198	156	508

Diese Kreuztabelle ist eine zweidimensionale Aggregation

Nimmt man noch andere Automodelle hinzu, kommt für jedes Modell eine weitere Ebene hinzu

Man erhält eine dreidimensionale Aggregation

Der CUBE-Operator

n-dimensionale Generalisierung der bisher genannten Konzepte

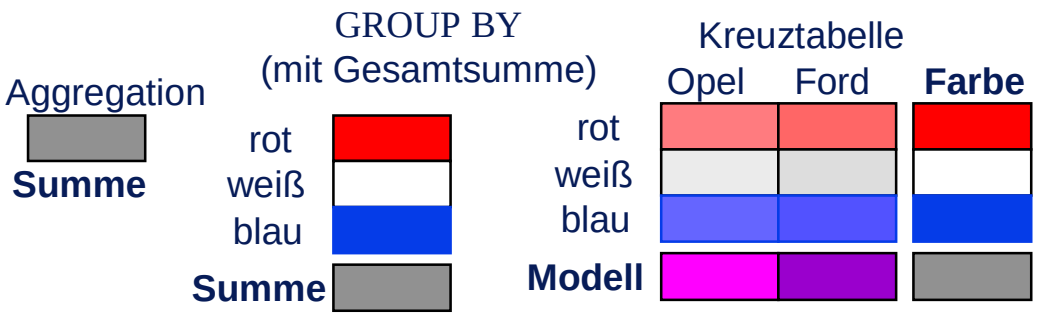
Der 0D Data Cube ist ein Punkt

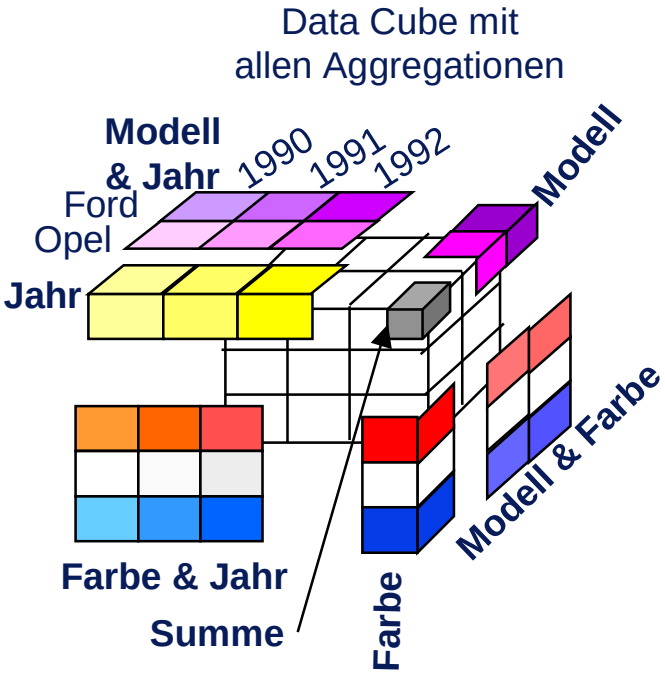
Der 1D Data Cube ist eine Linie mit einem Punkt

Der 2D Data Cube ist eine Kreuztabelle

Der 3D Data Cube ist ein Würfel mit drei sich überschneidenden Kreuztabellen

(Gray, Chaudhuri, Bosworth, Layman 1997)





Der CUBE-Operator

- Beispiel: **SELECT** Modell, Jahr, Farbe, **SUM**(Anzahl)
  FROM Autoverkäufe
  GROUP BY CUBE Modell, Jahr, Farbe
- Der Cube-Operator erzeugt eine Tabelle, die sämtliche Aggregationen enthält
- Es werden GROUP BYs für alle möglichen Kombinationen der Attribute berechnet
- Die Erzeugung der Tabelle erfordert die Generierung der Potenzmenge der zu aggregierenden Spalten.
- Bei n Attributen werden 2^n GROUP BYs berechnet
- Sei $C_1, C_2, ..., C_n$ die Kardinalität der n Attribute, dann ist die Kardinalität der resultierenden Data Cube-Relation $\prod(C_i + 1)$

Data Cube des Beispiels

Modell	Jahr	Farbe	Anzahl
Opel	1990	rot	5
Opel	1990	weiß	87
Opel	1990	blau	62
Opel	1991	rot	54
Opel	1991	weiß	95
Opel	1991	blau	49
Opel	1992	rot	31
Opel	1992	weiß	54
Opel	1992	blau	71
Ford	1990	rot	64
Ford	1990	weiß	62
Ford	1990	blau	63
Ford	1991	rot	52
Ford	1991	weiß	9
Ford	1991	blau	55
Ford	1992	rot	27
Ford	1992	weiß	62
Ford	1992	blau	39

Modell	Jahr	Farbe	Anzahl
Opel	1990	rot	5
Opel	1990	weiß	87
Opel	1990	blau	62
Opel	1990	ALL	154
Opel	1991	rot	54
Opel	1991	weiß	95
Opel	1991	blau	49
Opel	1991	ALL	198
Opel	1992	rot	31
Opel	1992	weiß	54
Opel	1992	blau	71
Opel	1992	ALL	156
Opel	ALL	rot	90
Opel	ALL	weiß	236
Opel	ALL	blau	182
Opel	ALL	ALL	508
Ford	1990	rot	64
Ford	1990	weiß	72
Ford	1990	blau	63
Ford	1990	ALL	189
Ford	1991	rot	52
Ford	1991	weiß	9
Ford	1991	blau	55
Ford	1991	ALL	116

Modell	Jahr	Farbe	Anzahl
Ford	1992	rot	27
Ford	1992	weiß	62
Ford	1992	blau	39
Ford	1992	ALL	128
Ford	ALL	rot	143
Ford	ALL	weiß	133
Ford	ALL	blau	157
Ford	ALL	ALL	433
ALL	1990	rot	69
ALL	1990	weiß	149
ALL	1990	blau	125
ALL	1990	ALL	343
ALL	1991	rot	106
ALL	1991	weiß	104
ALL	1991	blau	104
ALL	1991	ALL	314
ALL	1992	rot	58
ALL	1992	weiß	116
ALL	1992	blau	110
ALL	1992	ALL	284
ALL	ALL	rot	233
ALL	ALL	weiß	369
ALL	ALL	blau	339
ALL	ALL	ALL	941

Implementationsalternativen

- Physische Materialisierung des gesamten Data Cube:
  - beste Antwortzeit
  - hoher Speicherplatzbedarf
- Keine Materialisierung:
  - jede Zelle wird nur bei Bedarf aus den Rohdaten berechnet
  - kein zusätzlicher Speicherplatz
  - schlechte Antwortzeit
- Materialisierung von Teilen des Data Cube:
  - Werte vieler Zellen sind aus Inhalt anderer Zellen berechenbar
  - diese Zellen nennt man „abhängige" Zellen
  - Zellen, die einen All-Wert enthalten, sind abhängig
  - Problem: Welche Zellen des Data Cube materialisieren?
  - Zellen des Data Cube entsprechen SQL Anfragen (Sichten)

Abhängigkeit von Sichten

Die Abhängigkeitsrelation $\leq$ zwischen zwei Anfragen Q_1 und Q_2

$Q_1 \leq Q_2$ gdw. Q_1 kann beantwortet werden, indem die Ergebnisse von Q_2 verwendet werden. Q_1 ist abhängig von Q_2

- Anfragen bilden einen Verband unter folgenden Voraussetzungen:
  1. $\leq$ ist eine Halbordnung und
  2. es gibt ein maximales Element (eine oberste Sicht)
- Der Verband wird durch eine Menge von Anfragen (Sichten) L und der Abhängigkeitsrelation $\leq$ definiert und mit $\langle L, \leq \rangle$ bezeichnet
- Ein Verband wird dargestellt durch einen Graphen, in dem die Anfragen die Knoten sind und $\leq$ die Kanten.

Auswahl von Sichten

- Optimierungsproblem, das unter folgenden Bedingungen gelöst werden soll:
  - Die durchschnittliche Zeit für die Auswertung der Anfragen soll minimiert werden.
  - Man beschränkt sich auf eine feste Anzahl von Sichten, die materialisiert werden sollen, unabhängig von deren Platzbedarf
- Das Optimierungsproblem ist NP-vollständig.
- Heuristiken für Appoximationslösungen: Greedy-Algorithmus
- Der Greedy-Algorithmus verhält sich nie zu schlecht:
  Man kann zeigen, dass die Güte mindestens 63% beträgt.
  (Harinayaran, Rajaraman, Ullman 1996)

Der Greedy Algorithmus

- Gegeben ein Verband mit Speicherkosten C(v) für jede Sicht v
- Annahme: Speicherkosten = Anzahl der Reihen in der Sicht
- Beschränkung auf k materialisierte Sichten
- Nach Auswahl einer Menge S von Sichten wird der Nutzen der Sicht v relativ zu S mit B(v, S) bezeichnet und wie folgt definiert:

  1. Für jede Sicht $w \leq v$ wird B_w berechnet:

     (a) Sei u die Sicht mit den geringsten Kosten in S, so dass $w \leq u$

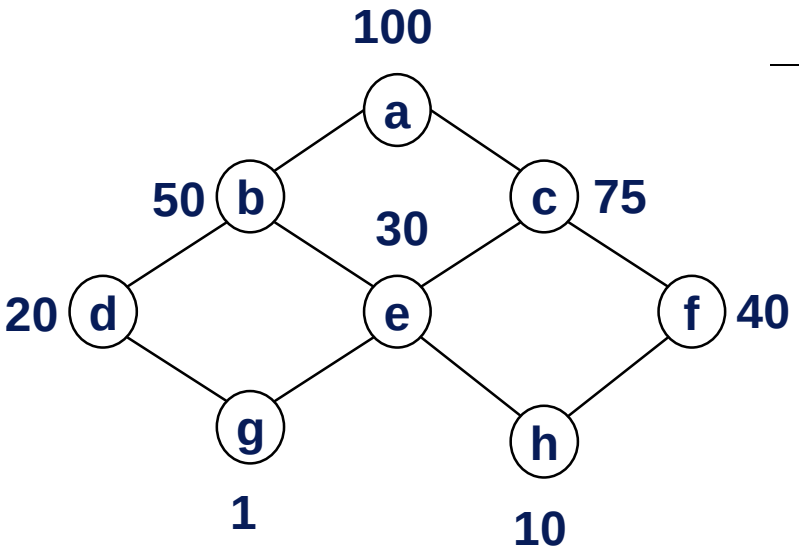
     (b) $B_w = \begin{cases} C(v) - C(u), & \text{falls } C(v) < C(u) \\ 0 & \text{ansonsten} \end{cases}$

  2. $B(v, S) = \sum_{w \leq v} B_w$

Der Greedy Algorithmus

```
1  S = {oberste Sicht}
2  for i = 1 to k do begin
3       Wähle die Sicht v∉ S, so dass B(v, S) maximal ist;
4       S = S ∪ {v}
5       end;
6  return S;
```

Beispiel



	Erste Wahl	Zweite Wahl	Dritte Wahl
b	50 x 5 = 250		
c	25 x 5 = 125	25 x 2 = 50	25 x 1 = 25
d	80 x 2 = 160	30 x 2 = 60	30 x 2 = 60
e	70 x 3 = 210	20 x 3 = 60	20 + 20 + 10 = 50
f	60 x 2 = 120	60 + 10 = 70	
g	99 x 1 = 99	49 x 1 = 49	49 x 1 = 49
h	90 x 1 = 90	40 x 1 = 40	30 x 1 = 30

Greedy Auswahl: b, d und f

Was wissen Sie jetzt?

- Möglichkeiten und Grenzen der Aggregation in SQL
- Einführung von Data Cubes zur Unterstützung von Aggregationen über n Dimensionen
- Implementationsalternativen von Data Cubes zur effizienten Anfragebearbeitung
- Greedy-Algorithmus zur Auswahl einer festen Anzahl von Sichten, die materialisiert werden

Lernen von Assoziationsregeln

Gegeben:

R eine Menge von Objekten, die binäre Werte haben

t eine Transaktion, $t \subseteq R$

r eine Menge von Transaktionen

Smin $\in$ [0,1] die minimale Unterstützung,

Confmin $\in$ [0,1] die minimale Konfidenz

Finde alle Regeln c der Form $X \rightarrow Y$, wobei $X \subseteq R$, $Y \subseteq R$, $X \cap Y = \{\}$

$$s(r,c) = \frac{\left|\left\{t \in r \middle| X \cup Y \in t\right\}\right|}{|r|} \geq s_{\min} \qquad conf(r,c) = \frac{\left|\left\{t \in r \middle| X \cup Y \in t\right\}\right|}{\left|\left\{t \in r \middle| X \in r\right\}\right|} \geq conf_{\min}$$

Binäre Datenbanken

R eine Menge von Objekten, die binäre Werte haben

A, B, C

r eine Menge von Transaktionen

t eine Transaktion, $t \subseteq R$

B,C

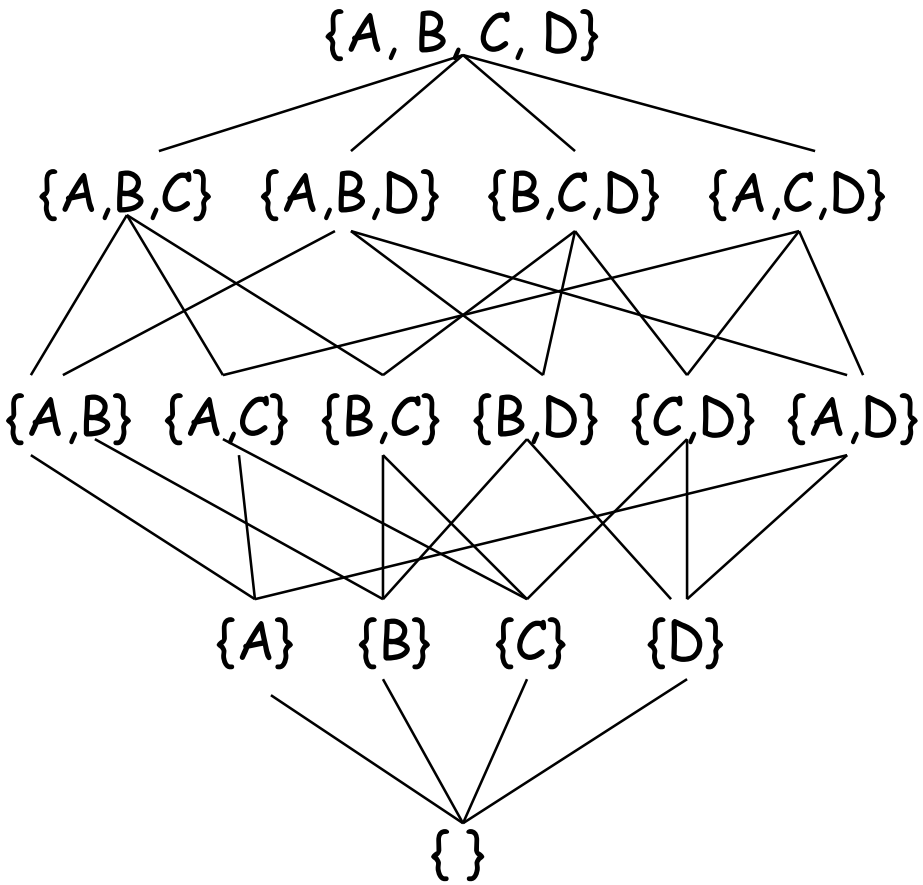
A	B	C	ID
0	1	1	1
1	1	0	2
0	1	1	3
1	0	0	4

Warenkorbanalyse

Aftershave	Bier	Chips	EinkaufsID
0	1	1	1
1	1	0	2
0	1	1	3
1	0	0	4

{Aftershave}→{Bier} $s = \frac{1}{4}$, conf = $\frac{1}{2}$
{Aftershave}→ {Chips} s = 0
{Bier} → {Chips} $s = \frac{1}{2}$, conf= 2/3 -- zusammen anbieten?
{Chips}→{Aftershave} s = 0
{Aftershave}→{Bier,Chips} s = 0

Wieder ein Verband...



Ordnungsrelation

- Hier ist die Ordnungsrelation die Teilmengenbeziehung.
- Eine Menge S_1 ist größer als eine Menge S_2, wenn $S_1 \supseteq S_2$.
- Eine kleinere Menge ist allgemeiner.

Assoziationsregeln

LH: Assoziationsregeln sind keine ILP-Regeln!

- In der Konklusion können mehrere Attribute stehen
- Attribute sind immer nur binär.
- Mehrere Assoziationsregeln zusammen ergeben kein Programm.

LE: Binärvektoren (Transaktionen)

- Attribute sind eindeutig geordnet.

Aufgabe:

- Aus häufigen Mengen Assoziationsregeln herstellen

Apriori Algorithmus

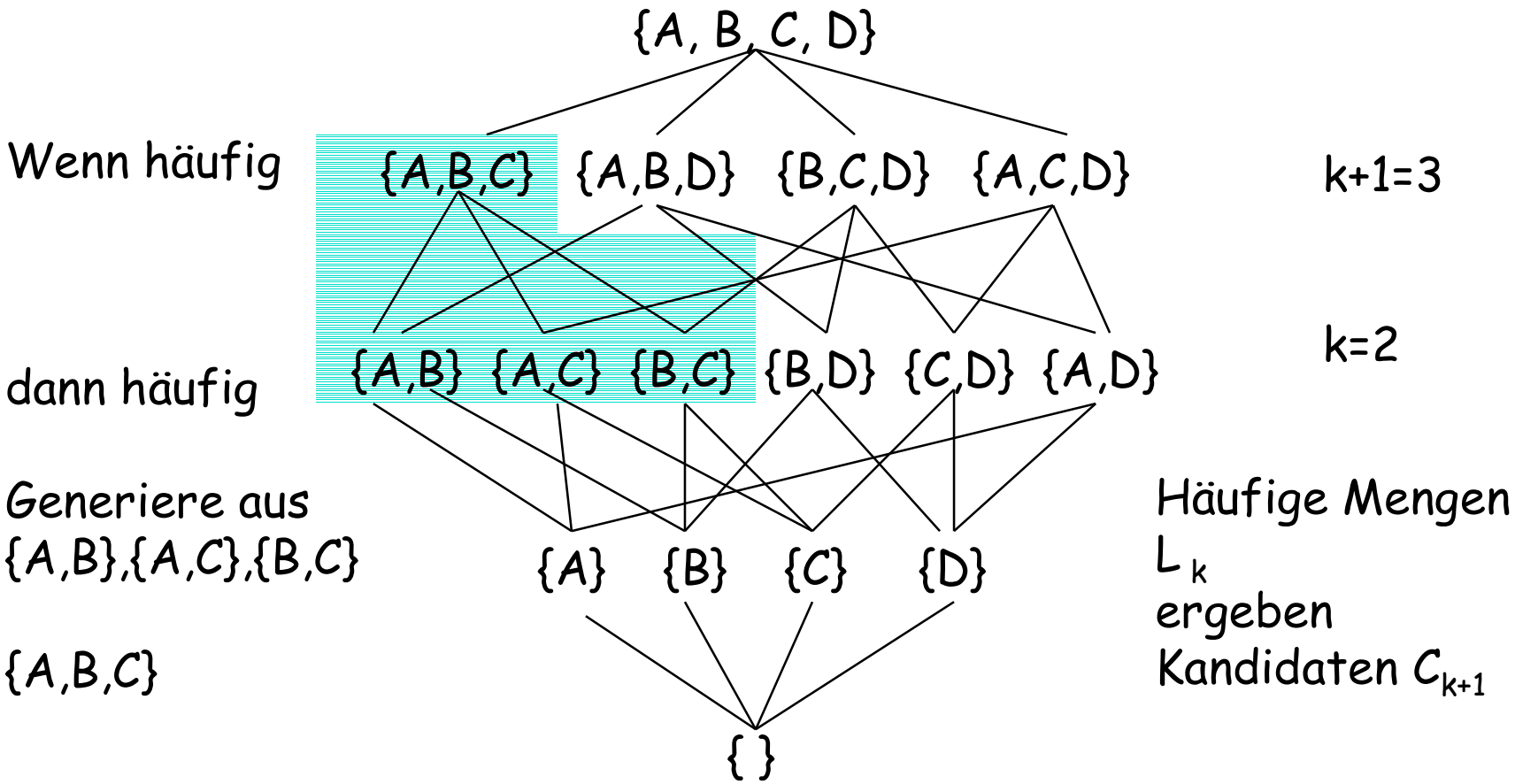
(Agrawal, Mannila, Srikant, Toivonen, Verkamo 1996)

LH des Zwischenschritts: Häufige Mengen $L_k = X \cup Y$
mit k Objekten (large itemsets, frequent sets)

- Wenn eine Menge häufig ist, so auch all ihre Teilmengen. (Anti-Monotonie)
- Wenn eine Menge selten ist, so auch all ihre Obermengen. (Monotonie)
- Wenn X in L_{k+1} dann alle $S_i \subseteq X$ in L_k (Anti-Monotonie)
- Alle Mengen L_k, die k-1 Objekte gemeinsam haben, werden vereinigt zu L_{k+1}.

Dies ist der Kern des Algorithmus', die Kandidatengenerierung.

Beispiel


{A, B, C, D}
Wenn häufig
{A,B,C}
{A,B,D}
{B,C,D}
{A,C,D}
k+1=3
dann häufig
{A,B}
{A,C}
{B,C}
{B,D}
{C,D}
{A,D}
k=2
Generiere aus
{A,B},{A,C},{B,C}
{A,B,C}
{A}
{B}
{C}
{D}
Häufige Mengen
L_k
ergeben
Kandidaten C_{k+1}
{ }

Beispiel

Gesucht werden Kandidaten mit k+1=5

L_4= { {ABCD}, {ABCE}, {ABDE}, {ACDE}, {BCDE} }

k-1 Stellen gemeinsam

vereinigen zu:

l = { ABCDE }

Sind alle k langen Teilmengen von l in L_4?

{ABCD} {ABCE} {ABDE} {ACDE} {BCDE} – ja!

Dann wird l Kandidat C_5.

L_4= { {ABCD}, {ABCE} }

l = { ABCDE }

Sind alle Teilmengen von l in L_4?

{ABCD} {ABCE} {ABDE} {ACDE} {BCDE} – nein!

Dann wird l nicht zum Kandidaten.

Kandidatengenerierung

Erzeuge-Kandidaten(L_k)

L_{k+1} := {}

Forall l_1, l_2 in L_k , sodass $l_1 = \{i_1, ..., i_{k-1}, i_k\}$

$l_2 = \{i_1, ..., i_{k-1}, i'_k\}$ $i'_k < i_k$

$l := \{i_1, ..., i_{k-1}, i_k, i'_k\}$

if alle k-elementigen Teilmengen von l in L_k sind

then $L_{k+1} := L_{k+1} \cup \{l\}$

Return L_{k+1}

Prune(C_{k+1}, r) vergleicht Häufigkeit von Kandidaten mit smin.

Häufige Mengen

Häufige-Mengen(R, r, smin)

$C_1 := \bigcup_{i \in R} \{i\}$, k=1,
L_1:= Prune(C_1)
while $L_k \neq \{\}$
  C_{k+1} := Erzeuge-Kandidaten(L_k)
  L_{k+1} := Prune(C_{k+1}, r)
  k:= k+1

Return $\bigcup_{j=2}^{k} L_j$

APRIORI

Apriori(R, s, smin, confmin)

L:= Häufige-Mengen(R, r, smin)

c:= Regeln (L, confmin)

Return c.

Regelgenerierung

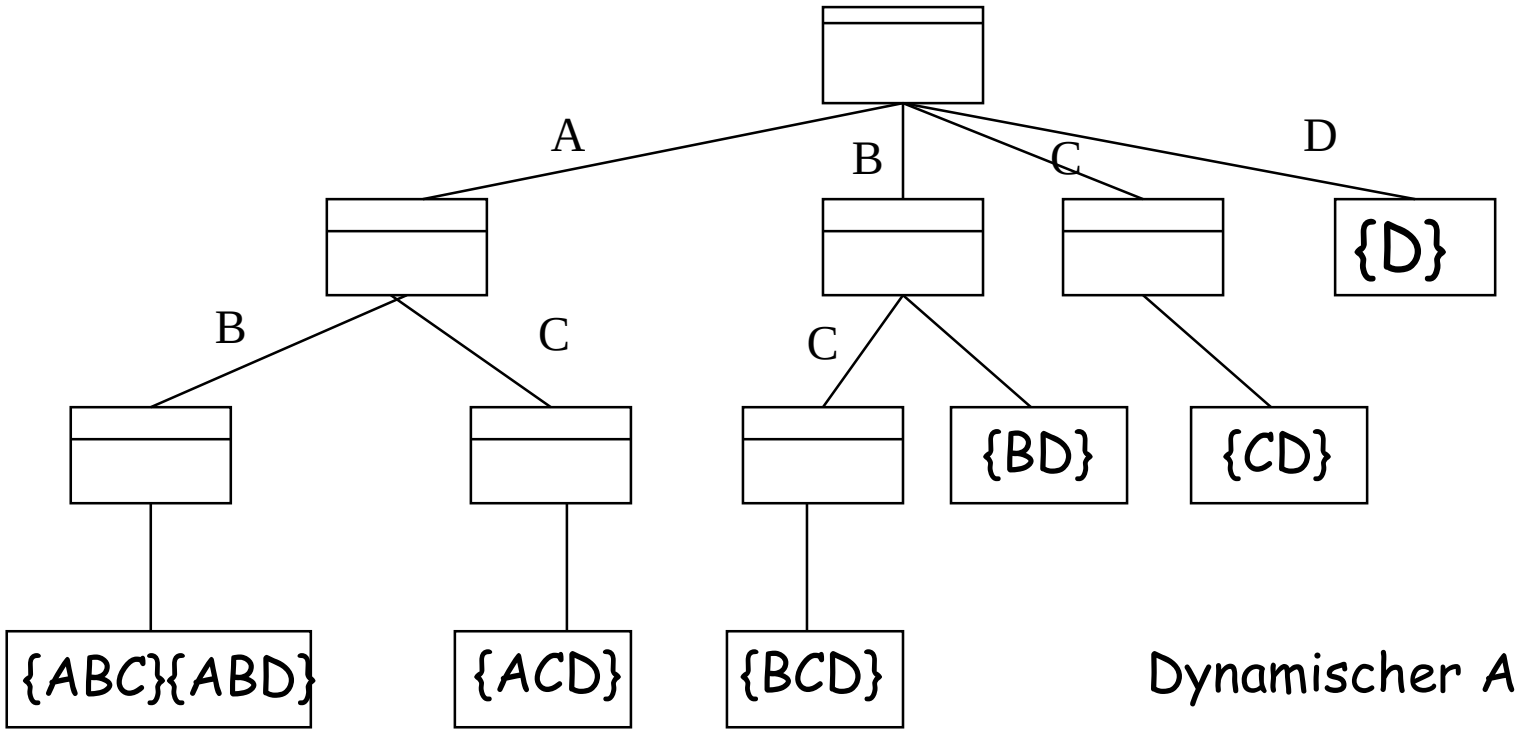
Aus den häufigen Mengen werden Regeln geformt.
Wenn die Konklusion länger wird, kann die Konfidenz sinken.
Die Ordnung der Attribute wird ausgenutzt:

$l_1 = \{i_1, \ldots, i_{k-1}, i_k\}$ $c_1 = \{i_1, \ldots, i_{k-1}\} \rightarrow \{i_k\}$ $conf_1$
$l_1 = \{i_1, \ldots, i_{k-1}, i_k\}$ $c_2 = \{i_1, \ldots\} \rightarrow \{i_{k-1}, i_k\}$ $conf_2$
...
$l_1 = \{i_1, \ldots, i_{k-1}, i_k\}$ $c_k = \{i_1\} \rightarrow \{\ldots, i_{k-1}, i_k\}$ $conf_k$

$conf_1 \geq conf_2 \geq \ldots \geq conf_k$

Implementierung

- Hash-Tree für den Präfixbaum, der sich aus der Ordnung der Elemente in den Mengen ergibt.
- An jedem Knoten werden Schlüssel und Häufigkeit gespeichert.



Dynamischer Aufbau

Was wissen Sie jetzt?

- Assoziationsregeln sind keine ILP-Regeln.
- Anti-Monotonie der Häufigkeit: Wenn eine Menge häufig ist, so auch all ihre Teilmengen.
- Man erzeugt häufige Mengen, indem man häufige Teilmengen zu einer Menge hinzufügt und diese Mengen dann auf Häufigkeit testet.
  Bottom-up Suche im Verband der Mengen.
- Monotonie der Seltenheit: Wenn eine Teilmenge selten ist, so auch jede Menge, die sie enthält.
- Man beschneidet die Suche, indem Mengen mit einer seltenen Teilmenge nicht weiter betrachtet werden.

Probleme von Apriori

- Im schlimmsten Fall ist Apriori exponentiell in R, weil womöglich alle Teilmengen gebildet würden.
  In der Praxis sind die Transaktionen aber spärlich besetzt.
  Die Beschneidung durch smin und confmin reicht bei der Warenkorbanalyse meist aus.
- Apriori liefert unglaublich viele Regeln.
- Die Regeln sind höchst redundant.
- Die Regeln sind irreführend, weil die Kriterien die apriori Wahrscheinlichkeit nicht berücksichtigen.
  Wenn sowieso alle Cornflakes essen, dann essen auch hinreichend viele Fußballer Cornflakes.

Aktuelle Forschung

- Kondensierte Repräsentationen
- Bessere Kriterien als support und Konfidenz
- Anfrageoptimierung im Sinne induktiver Datenbanken durch constraints

- Hier sehen wir nur die ersten beiden Verbesserungen.
- Die Konferenzen KDD, PKDD und ICDM sind aber voll von Beiträgen zu „frequent itemsets".

Kondensierte Repräsentationen

Ersetzen der Datenbank bzw. der Baumstruktur durch eine kondensierte Repräsentation,

- die kleiner ist als die ursprüngliche Repräsentation und
- aus der wir alle häufigen Mengen und ihre Häufigkeit ableiten können, ohne noch mal die Daten selbst anzusehen.

Kondensierte Repräsentationen für Assoziationsregeln:

- Closed item sets
- Free sets

Operator, der die Menge aller Assoziationsregeln ableitet:

- Cover operator

In anderen Worten:

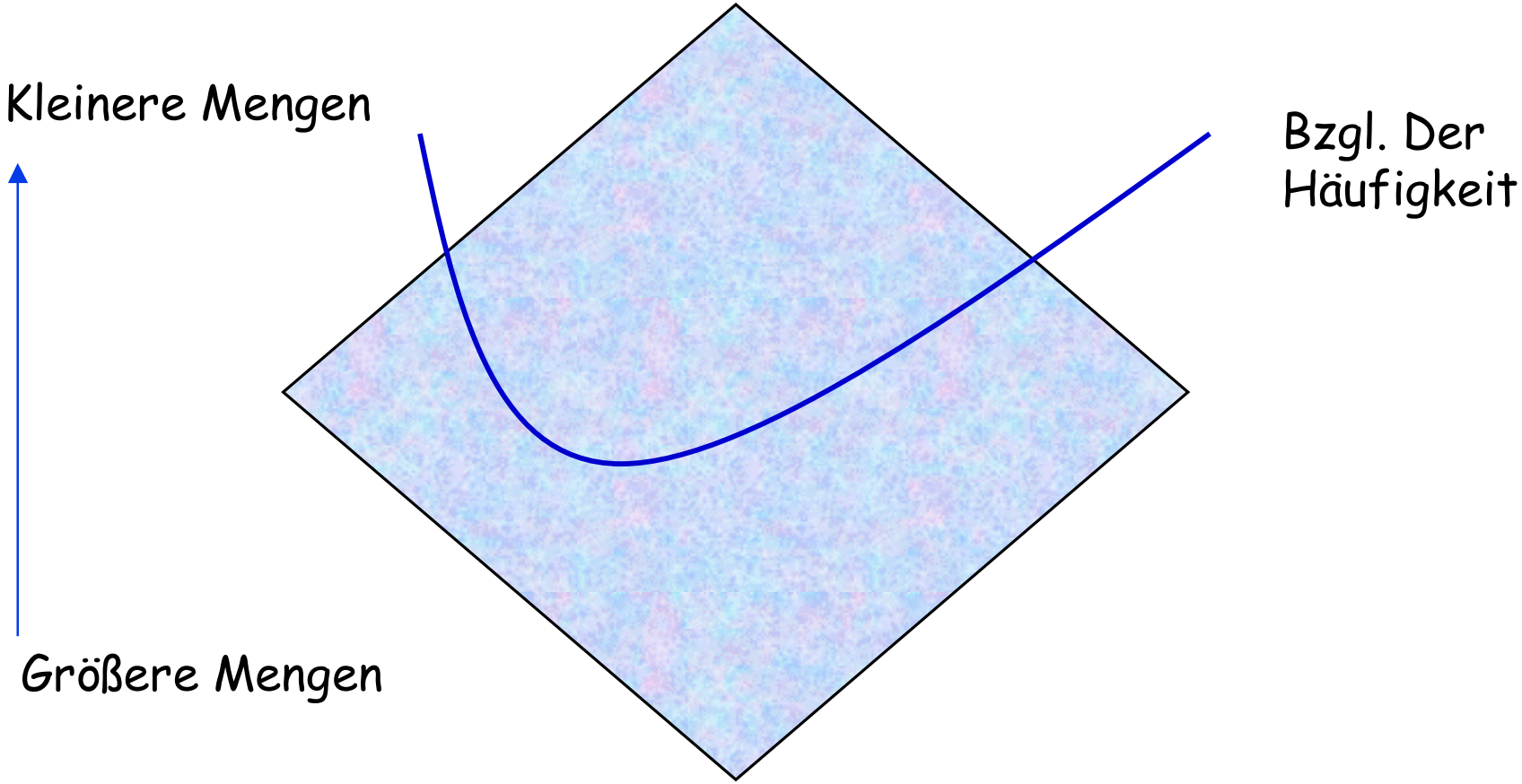
Wir hätten gern einen Versionenraum!
Der Versionenraum ist kleiner als der Hypothesenraum.
Außerhalb des Versionenraums kann das Lernziel nicht liegen.

Wir müssen also aus den Beispielen
- eine untere Grenze und
- eine obere Genze konstruieren.

Eine Halbordnung bzgl. Teilmengenbeziehung haben wir schon.

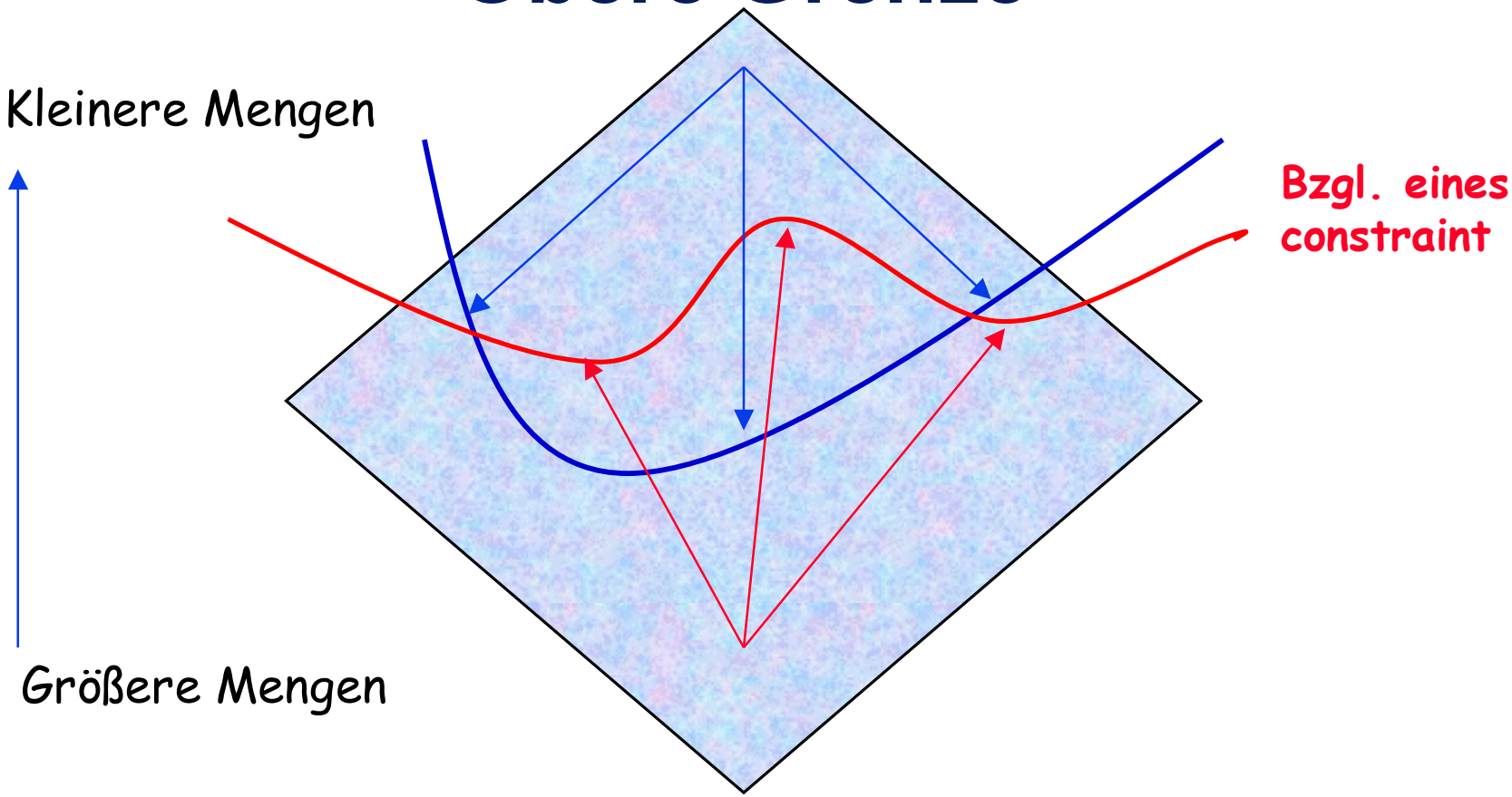
Die Grenzen haben wir auch.
Gemerkt?

Untere Grenze



- Wenn eine Menge häufig ist, so auch all ihre Teilmengen. (Anti-Monotonie)
- Beschneiden der Ausgangsmengen für die Kandidatengenerierung gemäß dieser Grenze!

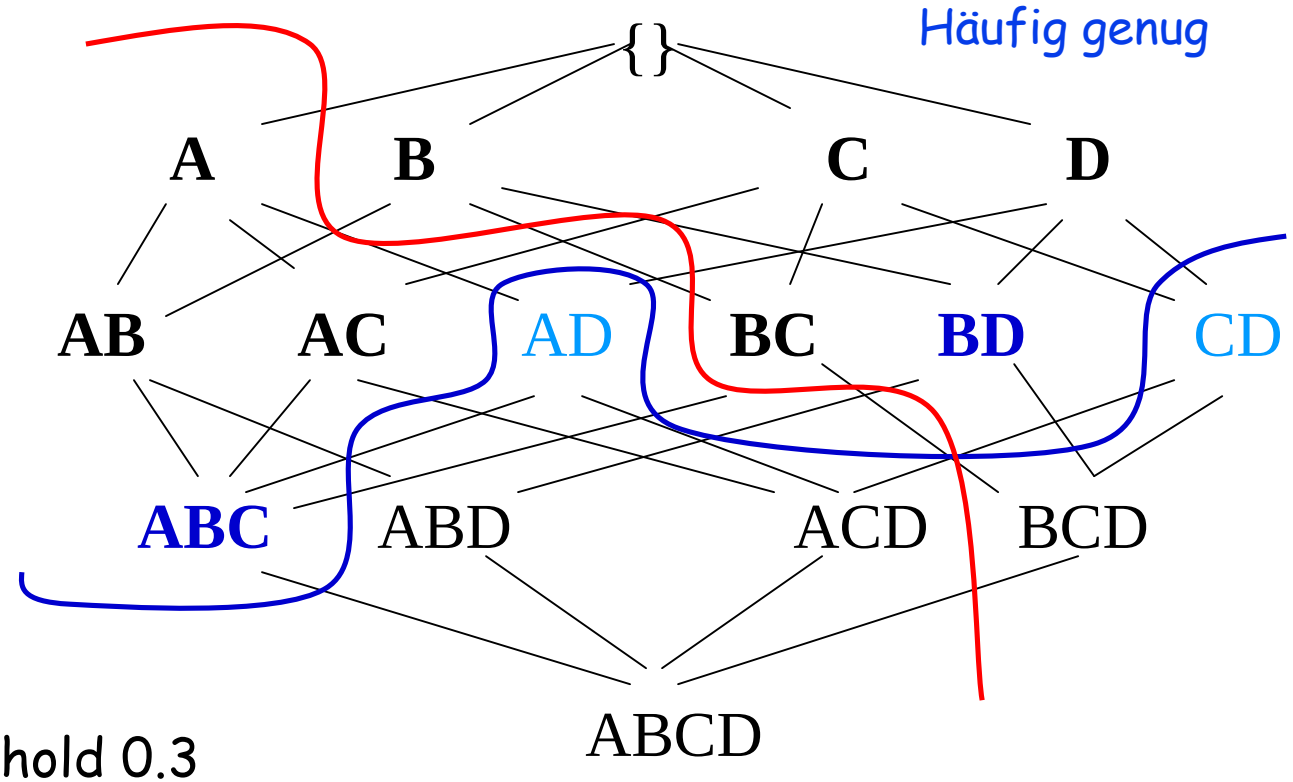
Obere Grenze



- Monotonie der Seltenheit: Wenn eine Teilmenge selten ist, so auch jede Menge, die sie enthält. Seltenheit ist ein constraint.
- Beschneidung der Kandidatengenerierung nach der Monotonie.

Beispiel

A	B	C	D
1	0	1	0
1	1	1	0
0	1	1	1
0	1	0	1
1	1	1	0



Frequency threshold 0.3

Dank an Jean-Francois Boulicaut!

Closed Item Sets

A	B	C	D
1	1	1	1
0	1	1	0
1	0	1	0
1	0	1	0
1	1	1	1
1	1	1	0

- closure(S) ist die maximale Obermenge (gemäß der Teilmengenbeziehung) von S, die noch genauso häufig wie S vorkommt.
- S ist ein *closed item set*, wenn closure(S)=S.
- Bei einem Schwellwert von 0,2 sind alle Transaktionen häufig genug.
- Closed sind: C, AC, BC, ABC, ABCD
  keine Obermenge von C kommt auch 6 mal vor;
  A kommt 5 mal vor, aber auch die Obermenge AC und keine Obermenge von AC
  ...

Kondensierte Repräsentation und Ableitung

Closed item sets sind eine kondensierte Repräsentation:

- Sie sind kompakt.
- Wenn man die häufigen closed item sets C berechnet hat, braucht man nicht mehr auf die Daten zuzugreifen und kann doch alle häufigen Mengen berechnen.

Ableitung:

- Für jede Menge S prüfen wir anhand von C:
  Ist S in einem Element X von C enthalten?
  - Nein, dann ist S nicht häufig.
  - Ja, dann ist die Häufigkeit von S ungefähr die von X.
    Wenn es in mehreren Elementen von C vorkommt, nimm die maximale Häufigkeit!

Freie Mengen (free sets)

- Eine Menge S ist frei, wenn es keine logische Regel (Konfidenz=1) zwischen ihren Elementen gibt, d.h.

$$\neg\exists X, Y \left| S = X \cup Y, Y \neq \{\ \}, X \Rightarrow Y \right.$$

- Eine Menge S ist δ-frei, wenn es keine Regel mit weniger als δ Ausnahmen zwischen ihren Elementen gibt.

- Die closed sets sind die closure der freien Mengen!
  Man kann die closed sets aus den freien Mengen berechnen.
- Freiheit ist eine anti-monotone Eigenschaft von Mengen.
  Deshalb kann man die freien Mengen effizient berechnen.

Beispiel

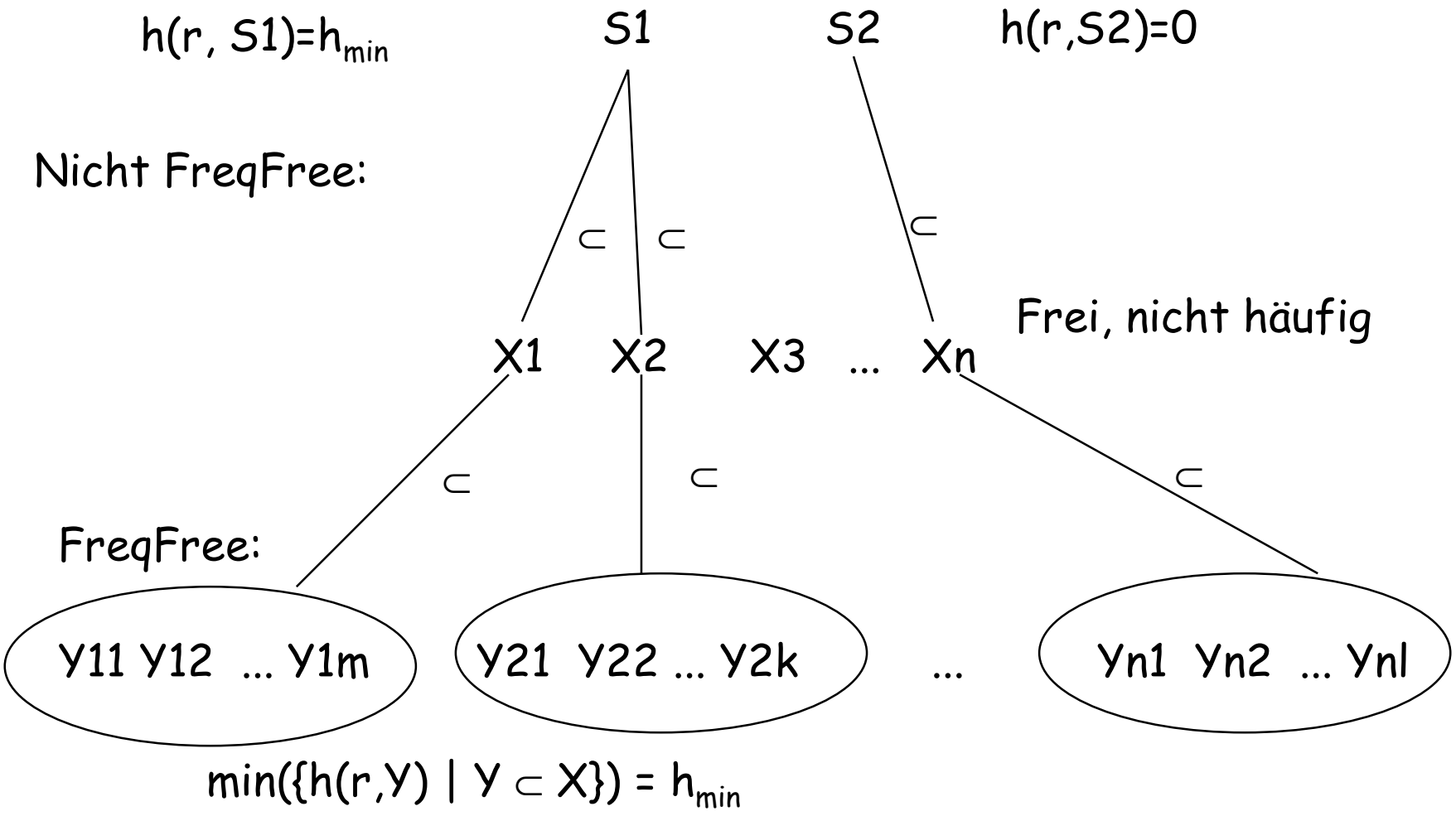
A	B	C	D
1	1	1	1
0	1	1	0
1	0	1	0
1	0	1	0
1	1	1	1
1	1	1	0

- Bei einem Schwellwert von 0,2 sind die häufigen freien Mengen:
  {}, A,B,D,AB
- Closed sind: C, AC, BC, ABC, ABCD
- Closure({})=C
  closure(A)=AC
  closure(B)= BC
  closure(D)=ABCD
  closure(AB)=ABC

Arbeiten mit freien Mengen

- Free(r, δ): Eine Menge X ist δ-frei, wenn es keine Regel zwischen ihren Elementen mit weniger als δ Ausnahmen gibt.
- Freq(r, σ): $\{X \mid X \subseteq R, |X \in r| / |r| \geq \sigma\}$
- FreqFree(r, σ, δ): Freq (r, σ) ∩ Free(r, δ)
- Negative Grenze Bd-(r, σ, δ): $\{X \mid X \subseteq R, X \notin$ FreqFree(r, σ, δ) und $\forall Y \subset X, Y \in$ FreqFree (r, σ, δ) $\}$
  Also die kürzesten Mengen, die gerade nicht häufig und frei sind, deren Teilmengen aber häufig und frei sind.
- Wir schätzen die Häufigkeit einer Menge S so ab:
  $\exists X \subseteq S$ und X ist δ-frei, aber nicht σ-häufig, dann nimm 0 als Häufigkeit von S.
  Sonst nimm die kleinste Anzahl im Vorkommen der Teilmengen X als Häufigkeit von S.

Abschätzung



MinEx

- Statt alle häufigen Mengen zu suchen, brauchen wir nur noch alle FreqFree(r, σ, δ) zu suchen.
- Bottom-up Suche im Halbverband der Mengen
  beginnt beim leeren Element, nimmt dann alle 1-elementigen Mengen,...
  endet bei den größten Mengen, die noch FreqFree(r, σ, δ) sind.
- Der Test, ob Mengen frei sind, erfordert das Bilden von strengen Regeln und erlaubt das Pruning der Mengen, in denen solche gefunden wurden.

Algorithmus von Jean-Francois Boulicaut

Algorithmus (abstrakt)

Gegeben eine binäre Datenbasis r über Objekten R und die Schwellwerte σ und δ,

Gebe FreqFree(r, σ, δ) aus.

1. C_0:={ {} }
2. i:=0
3. **While** $C_i \neq \{\}$ **do**
4. FreqFree$_i$:= {X |X $\in$ C_i, X ist σ-häufig und δ-frei}
5. C_{i+1}:= {X | X $\subseteq$ R, $\forall$Y $\subset$ X, Y $\in$ FreqFree$_j$ (r, σ, δ), j $\leq$ i }\ $\cup_{j \leq i} C_j$
6. i:=i+1 **od**
7. **Output** $\cup_{j < i}$ FreqFree$_j$

Pruning

- In der i-ten Iteration werden die δ–starken Regeln der Form $X \rightarrow \{A\}$ berechnet, wobei
  X häufig und frei ist auf der i-ten Ebene und
  $A \subseteq R \backslash X$.
- Das Ergebnis wird verwendet, um alle nicht δ -freien Mengen zu entfernen – sie sind keine Kandiaten mehr in der i+1-ten Iteration.

Eigenschaften von MinEx

- Der Algoritmus ist immer noch aufwändig, aber schneller als APRIORI und schneller als die Verwendung von closed sets.
- Der Algorithmus ist exponentiell in der Menge .
- Der Algorithmus ist linear in der Menge der Datenbanktupel, wenn δ im selben Maße steigt wie die Zahl der Tupel.
  Wir verdoppeln δ, wenn wir die Tupelzahl verdoppeln.
- Der Algorithmus approximiert das „wahre" Ergebnis.
  In der Praxis ist eine Abweichung von 0,3% aber kein Problem.

Was wissen Sie jetzt?

- Es gibt zwei Repräsentationen, die weniger Elemente für eine Suche nach häufigen Mengen ausgeben als eben alle häufigen Mengen. Aus diesen Repräsentationen können alle häufigen Mengen hergeleitet werden.
  - Die closed sets sind maximale Obermengen von S mit derselben Häufigkeit wie S.
  - Die free sets sind Mengen, aus denen man keine Assoziationsregeln machen kann.
- Wenn man die häufigen freien Mengen berechnet, hat man die untere Grenze im Versionenraum für Assoziationsregeln gefunden.
- Der Algorithmus MinEx findet diese Grenze.

Prinzipien für Regelbewertungen

1. RI(A→ B) = 0, wenn |A → B| = (|A| | B|) /|r|
   A und B sind unabhängig.
2. RI(A→ B) steigt monoton mit |A → B|.
3. RI(A→ B) fällt monoton mit |A| oder |B| .

Also: RI > 0, wenn |A → B| > (|A| | B|) /|r|
d.h., wenn A positiv mit B korreliert ist.

RI < 0, wenn |A → B| > (|A| | B|) /|r|
d.h., wenn A negativ mit B korreliert ist.

Wir wissen, dass immer |A → B| ≤ |A| ≤ | B| gilt, also
RImin wenn |A → B| = |A| oder |A| = | B|

RImax wenn |A → B| = |A| = | B|

Piatetsky-Shapiro 1991

Konfidenz

- Die Konfidenz erfüllt die Prinzipien nicht! (Nur das 2.)
  Auch unabhängige Mengen A und B werden als hoch-konfident bewertet.
- Die USA-Census-Daten liefern die Regel
  aktiv-militär → kein-Dienst-in-Vietnam mit 90% Konfidenz.
  Tatsächlich ist s(kein-Dienst-in-Vietnam)=95%
  Es wird also wahrscheinlicher, wenn aktiv-militär gegeben ist!
- Gegeben eine Umfrage unter 2000 Schülern, von denen 60% Basketball spielen, 75% Cornflakes essen. Die Regel
  Basketball → Cornflakes hat Konfidenz 66%
  Tatsächlich senkt aber Basketball die Cornflakes Häufigkeit!

Signifikanztest

- Ein einfaches Maß, das die Prinzipien erfüllt, ist:

$$|A \to B| - \frac{|A||B|}{|r|}$$

- Die Signifikanz der Korrelation zwischen A und B ist:

$$\frac{|A \to B| - {|A||B|}/{|r|}}{\sqrt{|A||B|\left(1 - {|A|}/{|r|}\right)\left(1 - {|B|}/{|r|}\right)}}$$

Sicherheitsmaß

Shortliffe, Buchanan 1990 führten ein Sicherheitsmaß CF (für Regeln in Wissensbasen) ein.

- Wenn $conf(A \rightarrow B) > s(B)$
  $CF(A \rightarrow B) = conf(A \rightarrow B) - s(B)/(1-s(B))$
- Wenn $conf(A \rightarrow B) < s(B)$
  $CF(A \rightarrow B) = conf(A \rightarrow B)$
- Sonst
  $CF(A \rightarrow B) = 0$.

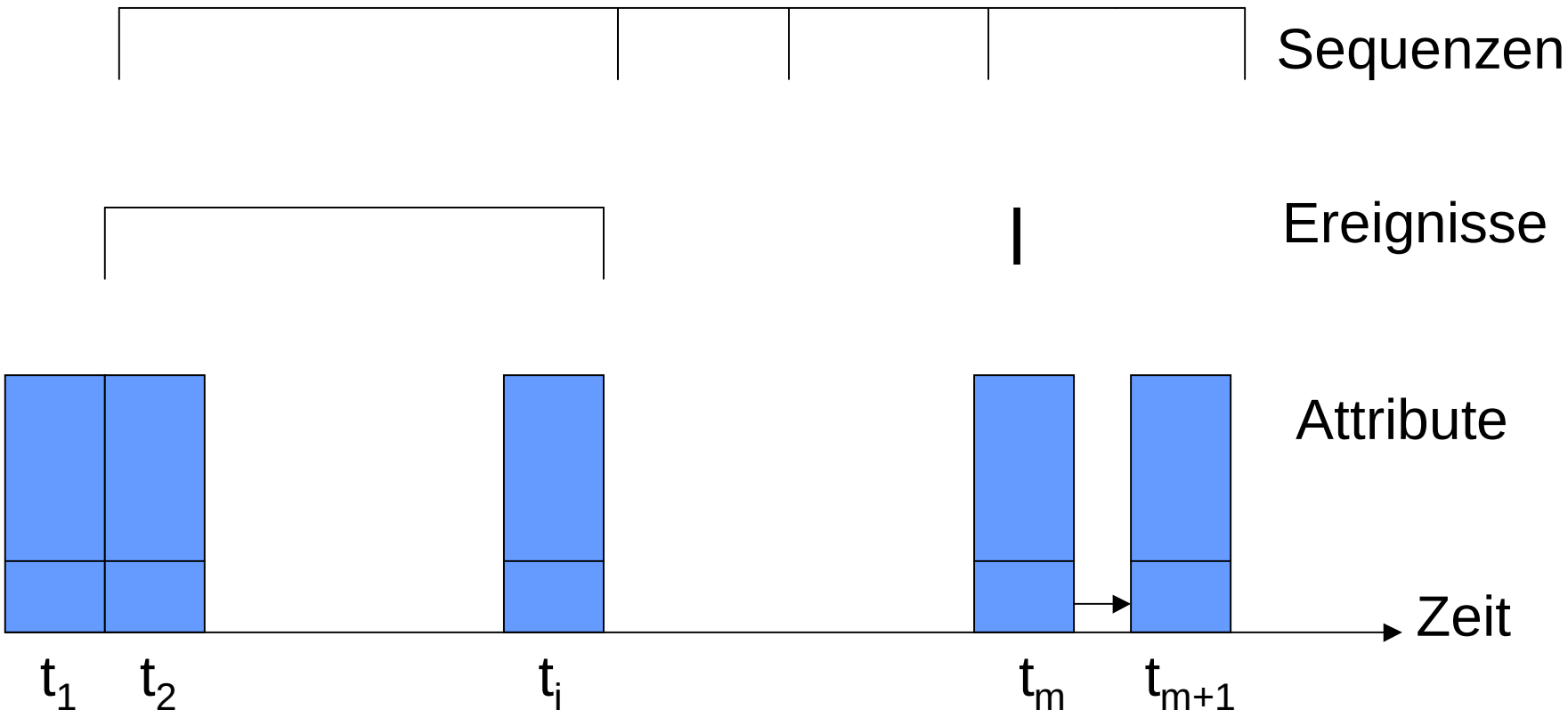
Das Sicherheitsmaß befolgt die Prinzipien für Regelbewertung.

Wendet man Signifikanztest oder Sicherheitsmaß an, erhält man weniger (irrelevante, irreführende) Assoziationsregeln.

Was wissen Sie jetzt?

- Sie haben drei Prinzipien für die Regelbewertung kennengelernt:
  - Unabhängige Mengen sollen mit 0 bewertet werden.
  - Der Wert soll höher werden, wenn die Regel mehr Belege hat.
  - Der Wert soll niediger werden, wenn die Mengen weniger Belege haben.
- Sie haben drei Maße kennengelernt, die den Prinzipien genügen:
  - Einfaches Maß,
  - statistisches Maß und
  - Sicherheitsmaß.

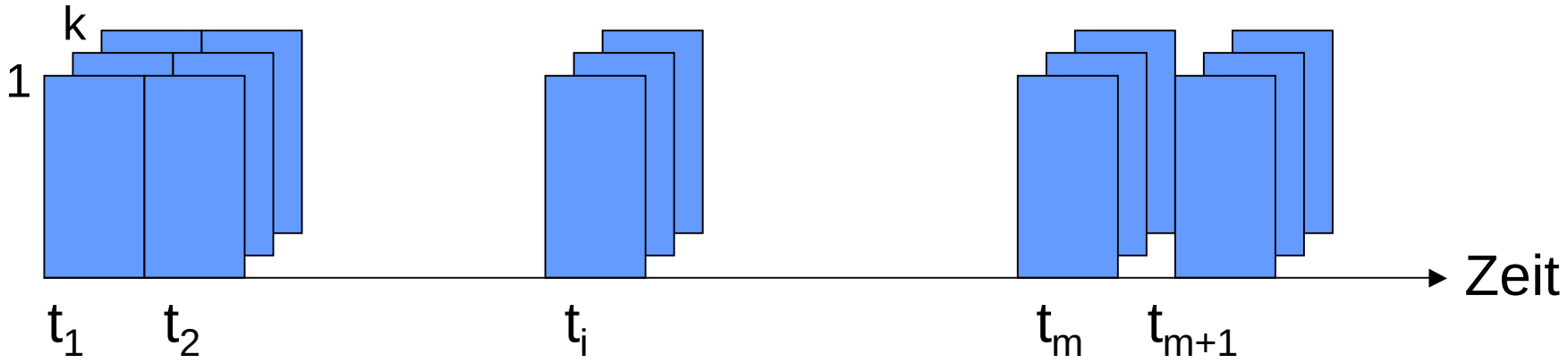
Zeitphänomene



Univariat - Multivariat

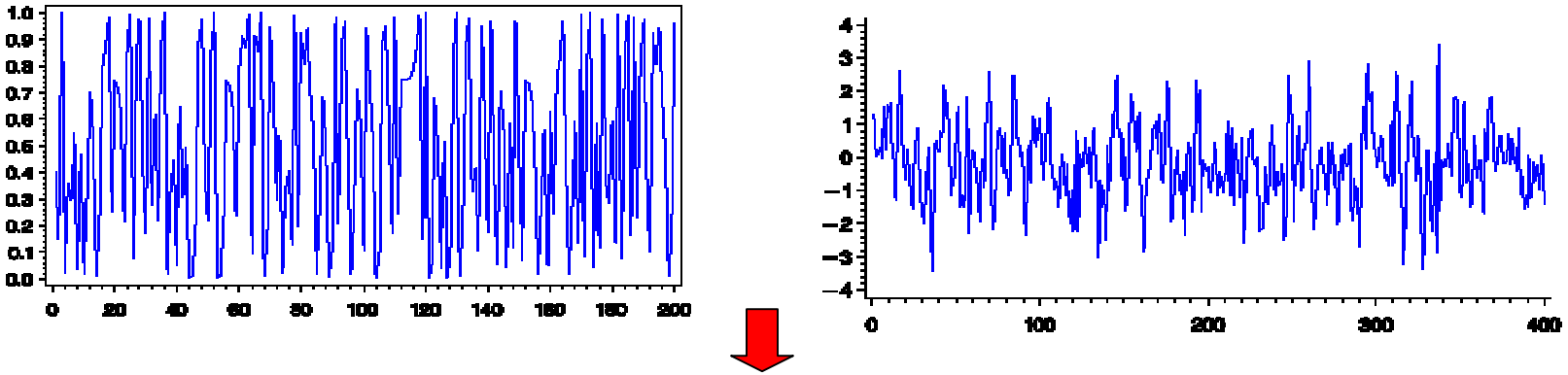
Univariat - ein Attribut pro Zeit (Herzfrequenz)

Multivariat - k Attribute (Herzfrequenz, Atemfrequenz, Blutdruck)



Beispiele für Zeitreihen

- Messwerte von einem Prozess
  - Intensivmedizin
  - Aktienkurse
  - Wetterdaten
  - Roboter



Kontinuierliche Messung in z.B. Tagen, Stunden, Minuten, Sekunden

Beispiele für Ereignisse

- Datenbankrelationen
  - Vertragsdaten, Verkaufsdaten, Benutzerdaten
  - Lebenssituation (Einkommen, Alter)

Verkäufe	Monat	Anzahl	Verkäufer	...
	Juni	256	Meier	...
	...	...	...	...

Ereignisse mit Zeitangaben in Jahren, Monaten, Tagen

Lernaufgaben

- Univariat
  - Vorhersagen der k+n-ten Beobachtung
  - einen allgemeinen Trend erkennen (alle Elemente steigen)
  - Lokale Trends finden (Zyklen, lokal steigende Werte)
  - Finde von einem Standard abweichende Werte (Ausreißer)
  - Clustering: Fasse ähnliche Bereiche von aufeinanderfolgen Werte zu Clustern zusammen
- Multivariat
  - Finde zusammen auftretende Werte

Repräsentation der Eingabedaten

Multivariat: i_l : $<t_1, a_{11}, \ldots, a_{1k}>$
$<t_2, a_{21}, \ldots, a_{2k}>$
...
$<t_i, a_{i1}, \ldots, a_{ik}>$

Univariat: i_l : $<t_1, a_1>$
$<t_2, a_2>$
...
$<t_i, a_i>$

Lernaufgaben

Lernaufgaben bei einer gegebenen Sequenz von Ereignissen:

(Menge von Ereignissen in partieller Ordnung)

1. Finde häufige Episoden in Sequenzen [Mannila et al.]
   - Wenn A auftritt, dann tritt B in der Zeit T auf [Das et al.]
2. Beziehungen zwischen Zeit-Intervallen lernen [Höppner]
   - A startet vor B, B und C sind gleich

Repräsentation der Eingabedaten

Ein Ereignis ist ein Tripel (Zustand, Start, Ende).

Der Zustand kann ein Wert oder ein Label (Trend bzw. eine Eigenschaft) sein.

Beispiele.:
(Steigend, 3, 5); (Fallend, 7, 9); (Stabil, 10, 14)

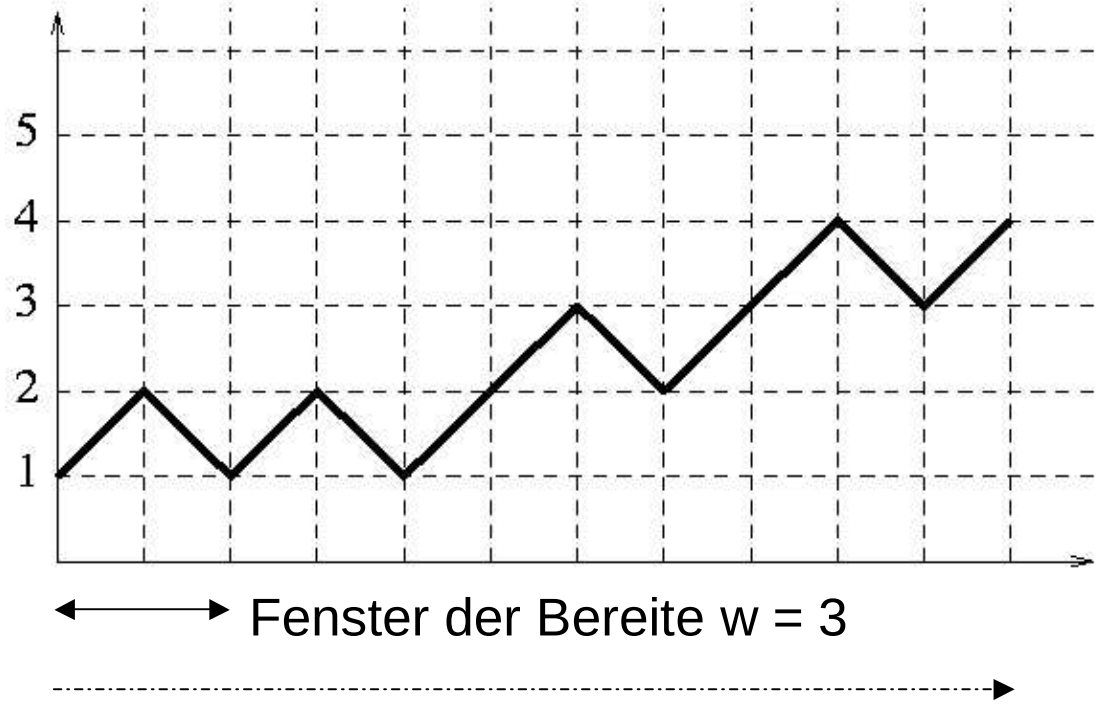
- Möglichkeiten der Darstellung
  1. Sequenz Vektor: $I : T_1A_1,\ldots, T_iA_i$
  2. Fakten: $P(I_1, T_b, T_e, A_r, \ldots A_s)$

Wie finde ich die Ereignisse in Zeitreihen?

- Fenster fester Länge w
  - vorgebende oder erlernte Muster
- Inkrementelle Analyse der Zeitreihe nach vorgegebenen Mustern [Morik/etal/99b]
  - Beispiel: Roboter
  - Vorteil: Dynamische Länge
- Diskretisierung durch Clustering [Das et al.]

Clustering Vorbereitung

Zeitreihe $s = (x_1,...,x_n)$ in Subsequenzen $s_i = (x_i,...,x_{i+w-1})$ aufteilen



⇨ Schritt 2

Clustering

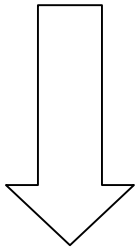
Distanzmaß $d(s_i,s_j)$: Entfernung zwischen zwei Subsequenzen
Bsp.: Euklidischer Abstand $(\Sigma(x_i-y_i)^2)^{0,5}$
Konstante $d > 0$: gibt an, wie groß der Unterschied zwischen den Subsequenzen sein darf

a1= 

a2= 

a3= 

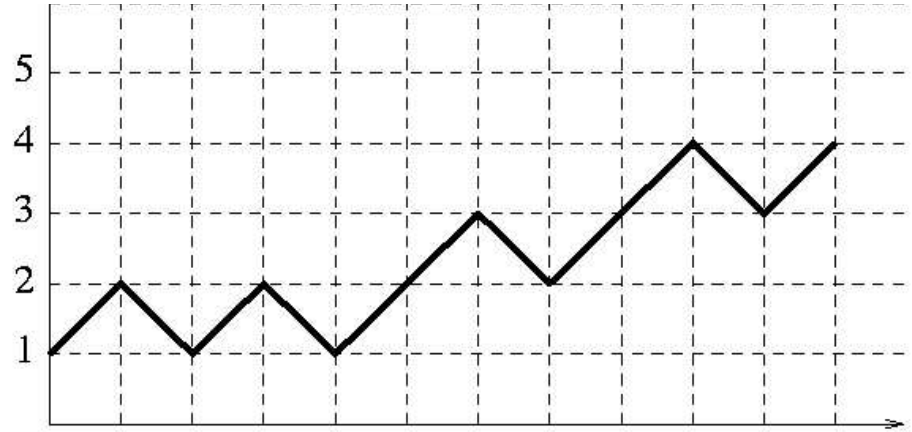
Bilde aus der Menge aller Subseqenzen
Cluster $C_1,...C_k$

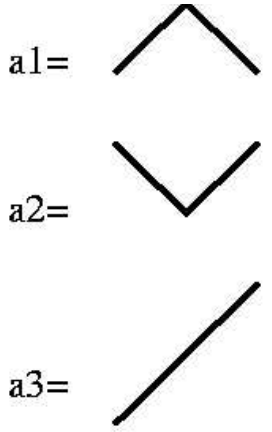


Jedes Cluster erhält ein Symbol $a_1,..a_k$ („Shapes")

Anwendung des Clustering

Die Serie $s = (x_1,\ldots,x_n)$ kann jetzt mit Hilfe der shapes beschrieben werden („diskretisiert")





Original time series = (1, 2, 1, 2, 1, 2, 3, 2, 3, 4, 3, 4)

Window width = 3

Discretized series = (a1, a2, a1, a2, a3, a1, a2, a3, a1, a2)

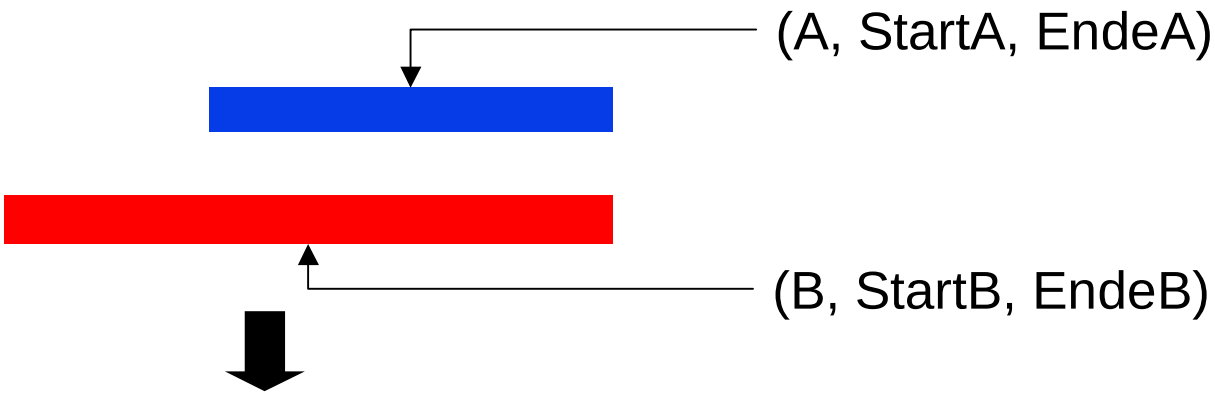
Primitive shapes after clustering

Regeln in diskreten Sequenzen

- Regeln der Form
  Wenn A auftritt, dann tritt B in der Zeit T auf einfach ableitbar mithhilfe APRIORI
- Berechnung in der Zeit $m*k^2$ möglich
  - (k=Anzahl der Symbole, m = #verschiedene Möglichkeiten für T)
- Erweiterung:
  - Wenn A_1 und A_2 und ... und A_h innerhalb der Zeit V auftritt, dann tritt B in der Zeit T auf
  - Microsoft $\downarrow$ (1), Microsoft $\uparrow$ (2) + Intel $\rightarrow$ (2) $\Rightarrow$ IBM $\rightarrow$ (3)
  - Problem: Anzahl der Regeln steigt stark an

Beziehungen zwischen Ereignissen

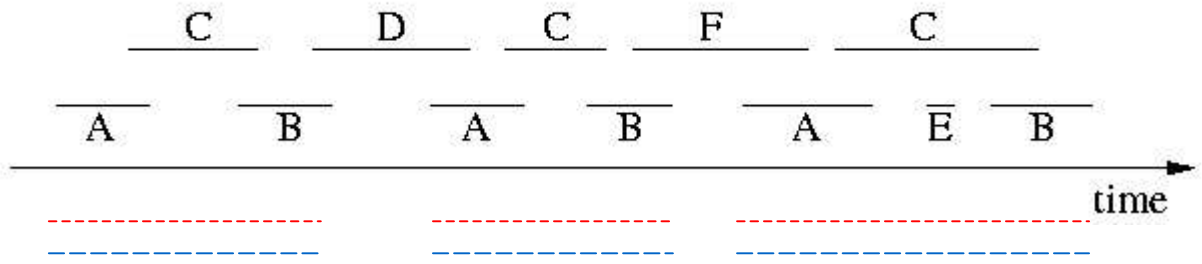
- Von James F. Allen wurden 13 verschiedene Intervallbeziehungen festgelegt:
  - A überlappt B, A beendet B, A vor B, A enthält B, ...
- Beispiel: A beendet B



StartB<StartA, EndeA = EndeB,

Beziehungen zwischen Zeit-Intervallen lernen [Höppner]

state interval sequence:



Darstellung der Beziehungen als Matrix:

R1

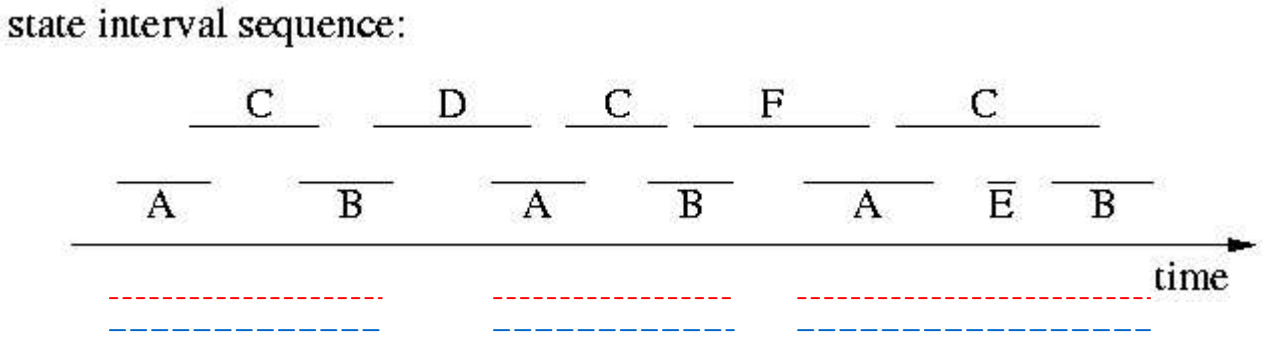
	A	B
A	=	b
B	a	=

R2

	A	B	C
A	=	b	o
B	a	=	io
C	io	o	=

(abbreviations: a=after, b=before, o=overlaps, io=is-overlapped-by)

Regeln



Die Regeln sind von der Form **P** $\rightarrow$ **R**

Prämisse P

	A	B
A	=	b
B	a	=

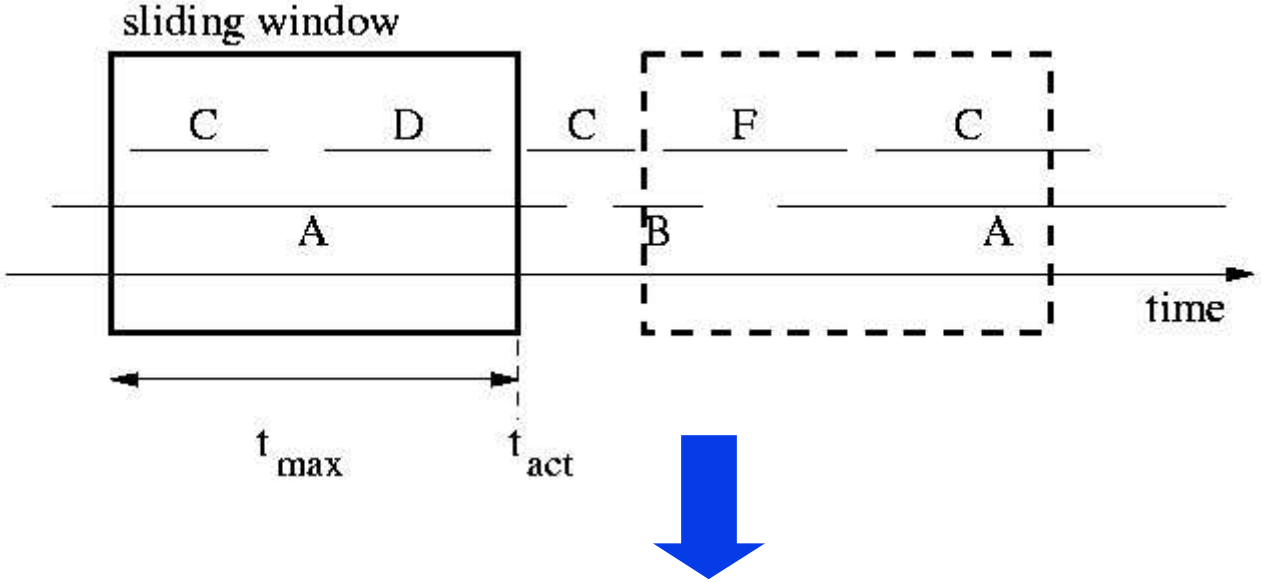
Regel R

	A	B	C
A	=	b	o
B	a	=	io
C	io	o	=

Beispiel: A, B, C sind Verträge verschiedener Kategorien

Häufige Muster finden

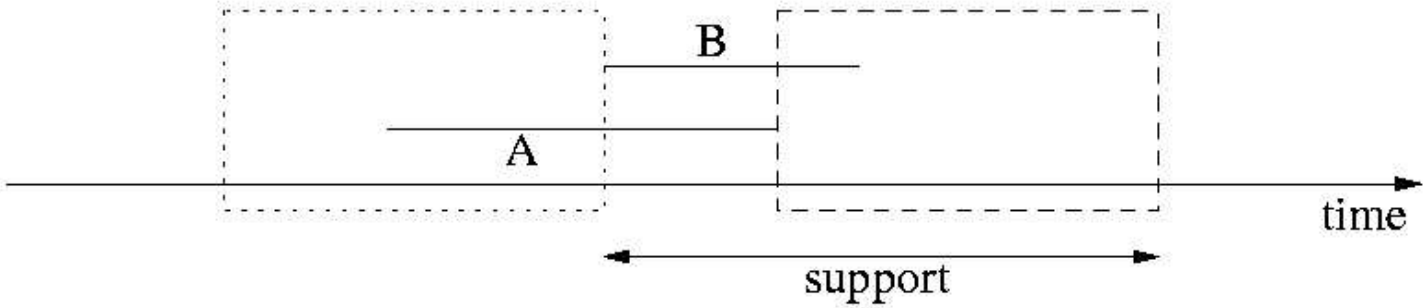
Muster muss im Fenster der Länge t_{max} beobachtbar sein



Der maximale Abstand zwischen den Ereignissen eines Muster ist begrenzt

Was bedeutet häufig?

Als Maß für die Häufigkeit von Mustern dient der „Support“



Ein Muster wird als häufig erachtet, wenn es einen Support > $supp_{min}$ hat

	A	B
A	=	o
B	io	=

Anwendung von APRIORI

- Ermittle den Support aller 1-Muster
- Im k-ten Lauf:
  - entferne alle Muster mit $supp < supp_{min}$
  - generiere aus den verbliebenen k-Mustern eine Menge von Kandidaten für k+1-Muster
  - ermittle den Support der Kandidaten im nächsten Lauf
- Wiederhole diese Schritte, bis keine häufigen Muster mehr gefunden werden können

- Generiere die Regeln aus den häufigen Mustern

Was wissen Sie jetzt?

- Man kann den Apriori Algorithmus für die Entdeckung von Zeitsequenzen anwenden.
- Der Ansatz von Gaudam Das et alii:
  - Fenster werden über die Zeitreihe geschoben
  - Die so erhaltenen Subsequenzen werden durch ein Distanzmaß ge-cluster-t. Es entstehen Muster wie aufsteigend, absteigend.
  - Mit den Mustern als Eingabe werden Assoziationsregeln gelernt.
- Der Ansatz von Frank Höppner:
  - Fenster werden über die Zeitreihe geschoben
  - Matritzen zu Allens Intervallen angelegt
  - Häufige, möglichst lange Sequenzen werden ermittelt und Assoziationsregeln gelernt.