Data Warehousing und Data Mining

Association Rule Mining

Ulf Leser

Wissensmanagement in der Bioinformatik

Inhalt dieser Vorlesung

- Association Rule Mining (ARM)
- A-Priori Algorithmus
- Varianten

Aufgabe

- Klassisches Problem: Warenkorbanalyse
- Transaktionen T mit binären Attributen I
  - Produkt (Item) gekauft oder nicht
- Frage: Welche Produkte werden häufig zusammen gekauft – was sind die „Frequent Itemsets“?
- Data Warehouse Kontext
  - Viele Millionen Transaktionen (T)
  - Viele tausend Produkte (I)
  - Man kann nicht die Häufigkeit von 2^I verschiedenen Itemsets in |T| Listen zählen

Formal

- Definition
  Gegeben eine Relation T von Transaktionen über Items der Menge I. Eine Assoziationsregel hat die Form (mit $0<n<|I|$, $i, i_1 .. i_n \in I$, $i \notin \{i_1, ..., i_n\}$)

$$\{i_1, ..., i_n\} \rightarrow i$$

- Intuition: Wer i_1, ..., i_n kauft, kauft auch i
- Real: Wir suchen nicht perfekte, sondern „gute“ Regeln
  - Sollen häufig greifen
  - Konsequenz sollen meistens gelten, wenn Antezedenz zutrifft

Support und Confidence

- Definition
  Sei R=H→i eine Assoziationsregel über T und I.
  - *Der Support von R, sup(R), ist die Häufigkeit von Tupeln in T, die die Items H∪i enthalten*

$$\sup(R) = |\{t \in T | H \cup i \in t\}|$$

  - *Die Confidence von R, conf(R), ist die Häufigkeit von H∪i in T normiert auf die Häufigkeit von H in T*

$$conf(R) = \frac{\sup(H \cup i)}{\sup(H)}$$

- Bemerkungen
  - Niedriger Support: Spezialregel, seltene Fälle
  - Niedrige Confidence: „Falsche“ Regel, gilt oft nicht

Beispiel

Transaction	Items
t_1	Bread,Jelly,PeanutButter
t_2	Bread,PeanutButter
t_3	Bread,Milk,PeanutButter
t_4	Beer,Bread
t_5	Beer,Milk

$X \Rightarrow Y$	s	α
Bread $\Rightarrow$ PeanutButter	60%	75%
PeanutButter $\Rightarrow$ Bread	60%	100%
Beer $\Rightarrow$ Bread	20%	50%
PeanutButter $\Rightarrow$ Jelly	20%	33.3%
Jelly $\Rightarrow$ PeanutButter	20%	100%
Jelly $\Rightarrow$ Milk	0%	0%

Quelle: Dunhum, Data Mining

Häufige Regeln

- ARM: Finde Regeln, deren Support größer ist als minsup
  - Also Regeln, die häufig greifen
- Beobachtung: Wenn $\{i_1,...i_m\}$ ein häufiges Itemset ist, dann ist jedes $\{i_1, ..., i_{j-1}, i_{j+1}, ..., i_m\} \rightarrow i_j$ eine häufige Assoziationsregel
- Also: Wenn man häufige Itemsets finden kann, kann man auch häufige Assoziationsregeln finden
- Damit ist noch nichts über die Confidence gesagt
  - Vorgehen: Häufige Regeln suchen, dann Confidence berechnen
  - Der zweite Schritt ist einfach: Division zweier Supports
- Bemerkung: Wenn $H=\{i_1, ..., i_n\} \rightarrow i$ häufig ist, dann ist $H' \rightarrow i$ mit $H' \subseteq H$ auch häufig - aber uninteressant
  - Closed ARM – finde nur interessante Regeln

Anwendungen

- Überall dort, wo Dinge zusammen passieren und man Zusammenhänge vermutet
  - Welche Aktienkurse steigen häufig an einer Börse gemeinsam?
  - Welche Gene werden häufig in einer Krankheit gemeinsam exprimiert?
  - Welche Ereignisse treten häufig vor einem Systemcrash in einem Cluster auf?
  - Welche Wörter treten häufig gemeinsam in bestimmten Dokumenten auf?
  - Was haben Studenten gemeinsam, die ihr Studium abbrechen?
  - ...
- Trick ist die Darstellung als Transaktionen über Items

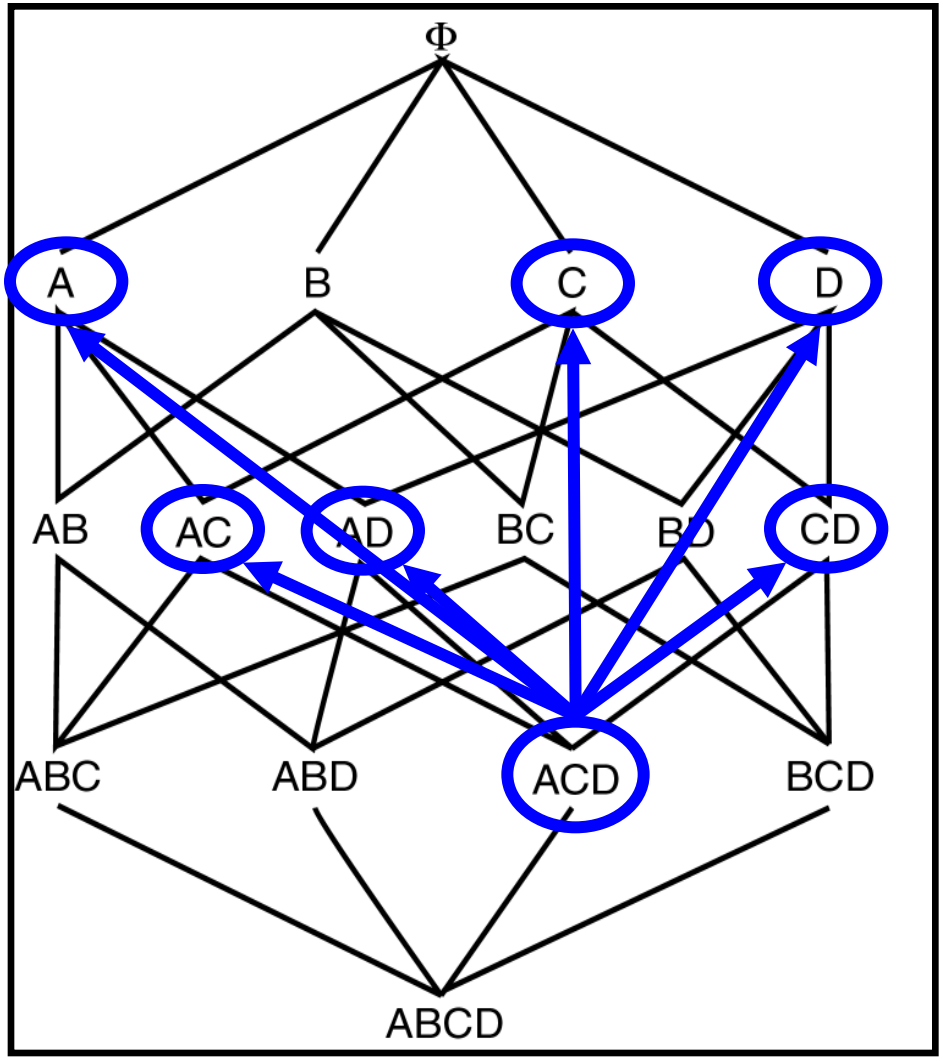
Inhalt dieser Vorlesung

- Association Rule Mining (ARM)
- A-Priori Algorithmus
  - Grundalgorithmus
  - Kandidatenerzeugung und –pruning
  - Datenstruktur für den Pruningschritt
  - Erweiterung: Partitionierung
  - Ableitung von Assoziationsregeln
- Varianten

Grundidee

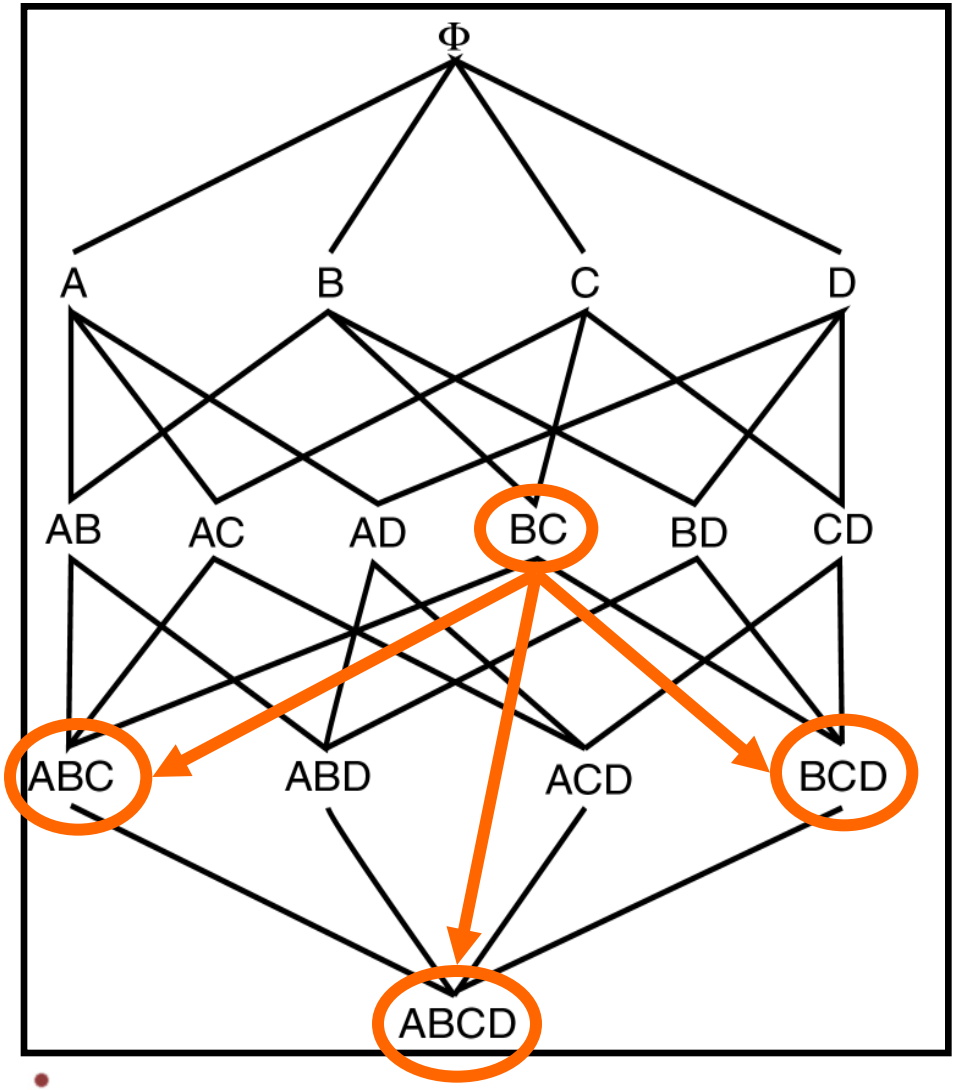
- A-Priori Regeln
  - Wenn $\{i_1,...i_m\}$ häufig ist, muss auch jedes $\{i_1, ..., i_j, i_{j+k}, ..., i_m\}$ häufig sein
    - Jede Teilmenge eines häufigen Itemsets ist häufig
  - Wenn $\{i_1,...i_m\}$ selten ist, muss auch jedes $\{i_1, ..., i_{m,} i_{m+1}, i_{m+k}\}$ selten sein
    - Jede Obermenge eines seltenen Itemsets ist selten
- Beweis: Einfach

Pruning mit A-Priori Regeln



- Upward
  - Wenn ACD häufig ist, muss AC, AD, CD, A, C, D häufig sein
  - ABCD kann häufig sein

Beispiel

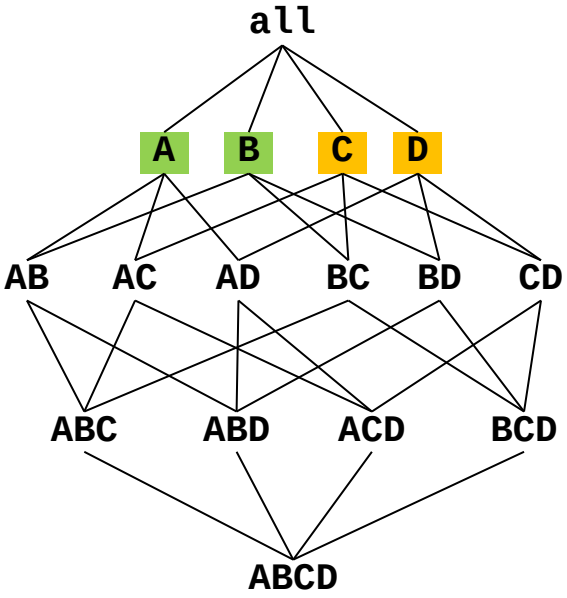


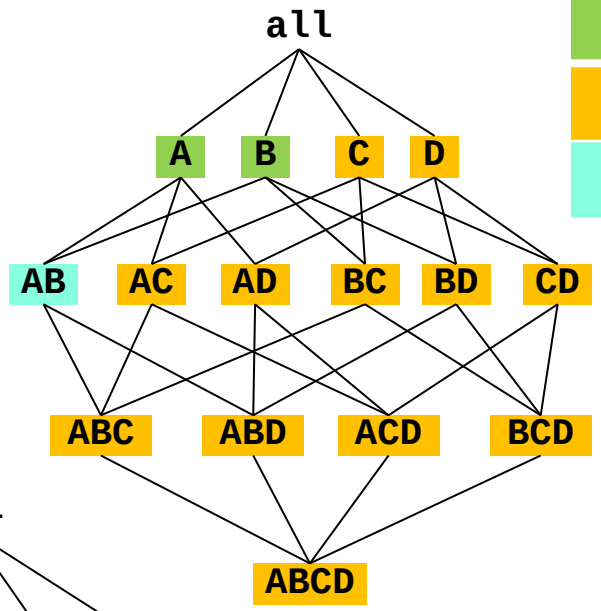
- Downward
  - Wenn ACD häufig ist, muss AC, AD, CD, A, C, D häufig sein
  - ABCD kann häufig sein
- Downward
  - Wenn BC selten ist, kann ABC, BCD und ABCD nicht häufig sein
  - B und C können selten sein

Grundidee [AS94]

- Wir wollen nicht 2^I Itemsets in T zählen
- A-Priori-Eigenschaft
  - Upward: Jede Teilmenge eines häufigen Itemsets ist häufig
  - Downward: Häufige große Itemsets bestehen nur aus häufigen kleinen Itemsets
- Beide Eigenschaften können wir zum zielgerichteten Aufzählen und Prunen verwenden
  - Starte mit kleinen Itemsets (einzelnen Items)
  - Aufzählen Downward: Nur Obermengen betrachten, die sich aus häufigen Teilmengen zusammensetzen lassen
  - Prunen Upward: Von diesen Obermengen können wir alle streichen, die eine seltene Teilmenge enthalten

Beispiel

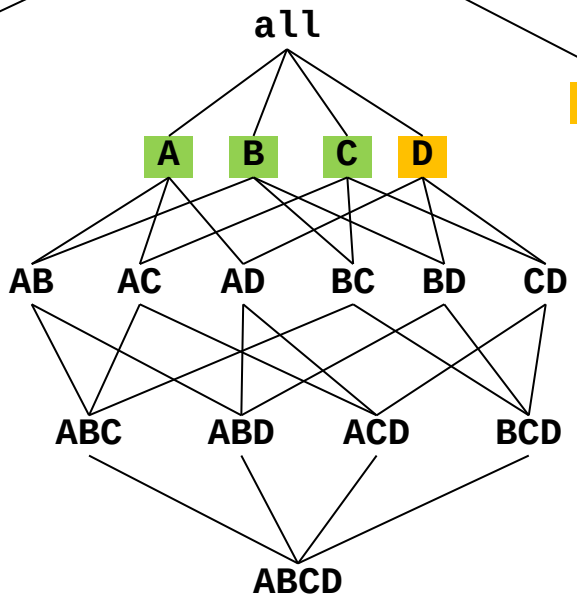


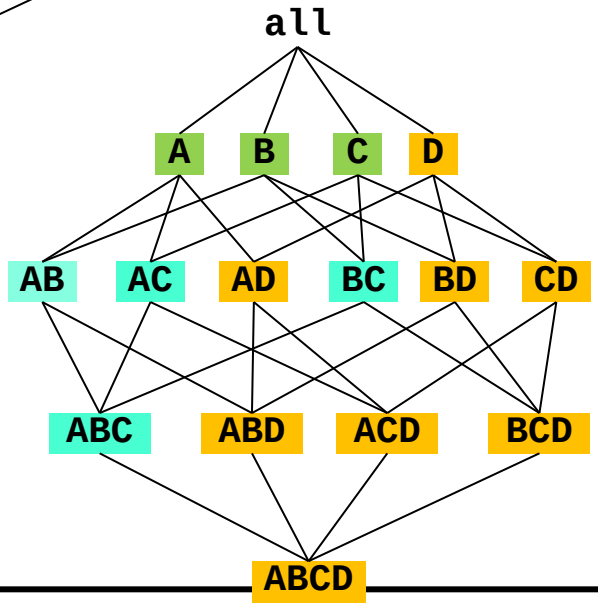


Surely frequent

Surely rare

Potentially frequent



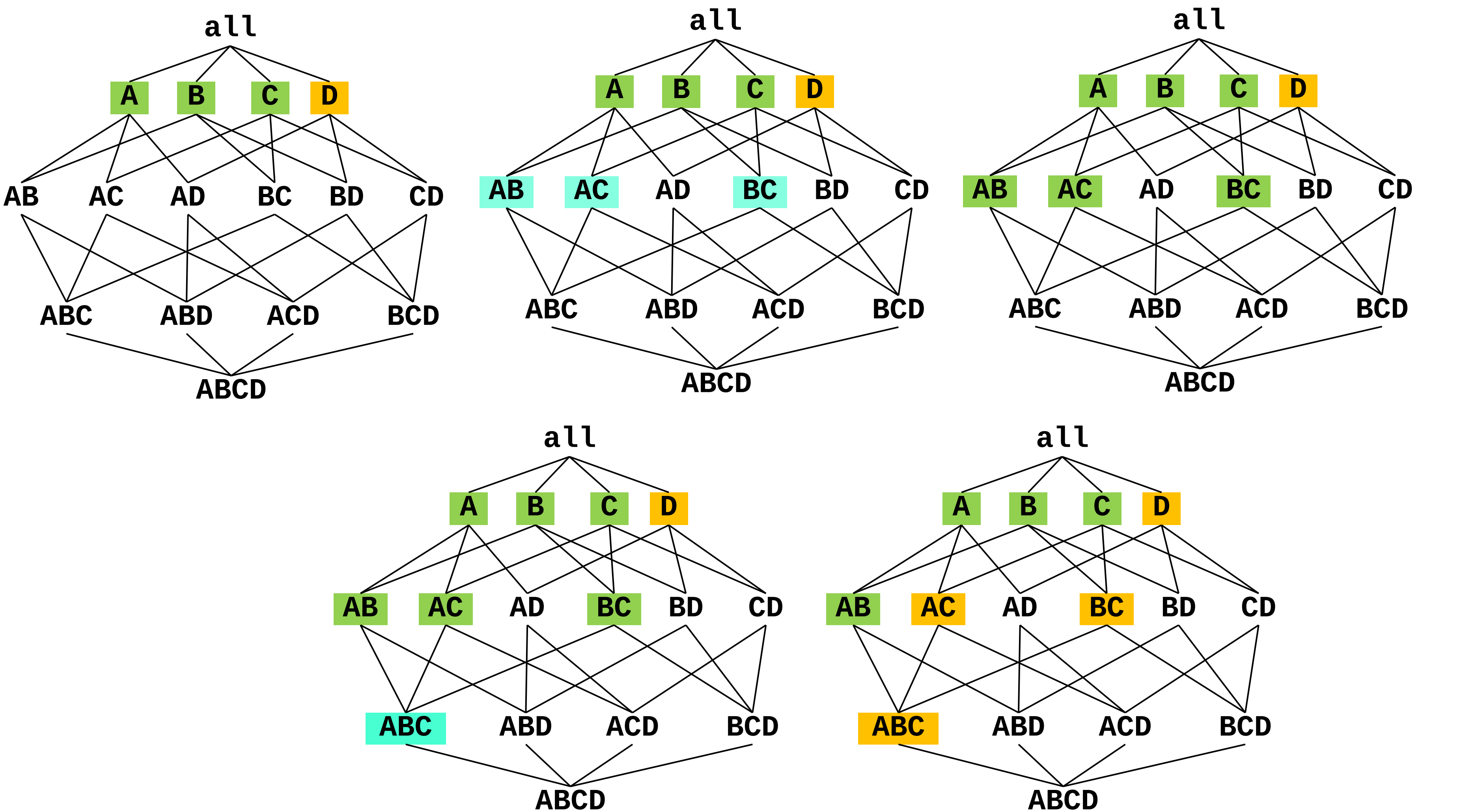


Top-Down Algorithmus

```
L1 := frequent 1-Itemsets aus I;          // Ein Scan von T
k := 2;                                   // k: Größe der nächsten Itemsets
while Lk-1≠∅ {                            // Noch Obermg möglich?
  Ck := generateCandidates( Lk-1);        // Alle Obermengen der Größe k
  for each t∈T                            // Scanne T
    C := contains( Ck,t);                 // Inkrementiere alle Ck, …
    for each c∈C {                        // …  die in t enthalten sind
      c.count++;
    }
  }
  Lk:={c∈Ck | (c.count≥minsup};   // Nur die häufigen merken
  k++;
}
return ∪Lk;
```

- Benötigt viele Scans der Datenbank
- Verbesserungen notwendig (und viele bekannt)

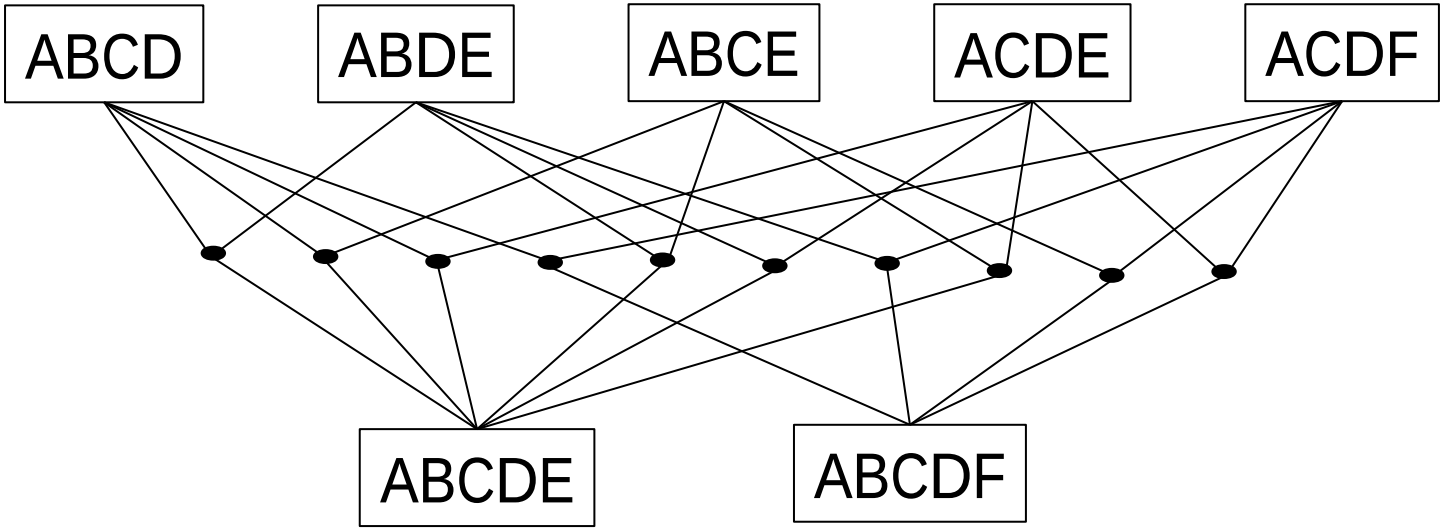
Beispiel


all
A B C D
AB AC AD BC BD CD
ABC ABD ACD BCD
ABCD
all
A B C D
AB AC AD BC BD CD
ABC ABD ACD BCD
ABCD
all
A B C D
AB AC AD BC BD CD
ABC ABD ACD BCD
ABCD
all
A B C D
AB AC AD BC BD CD
ABC ABD ACD BCD
ABCD
all
A B C D
AB AC AD BC BD CD
ABC ABD ACD BCD
ABCD

Kandidatengenerierung

- Gegeben: Menge L_{k-1} aller häufigen Itemsets mit k-1 Elementen
- generateCandidates: Finde alle Itemsets mit k Elementen, die nur häufige Teilmengen enthalten
- Erster Versuch: Bilde alle Obermengen der Größe k, die sich aus der Union von zwei Mengen aus L_{k-1} ergeben
  - Problem 1: Viele Obermengen werden mehrmals gebildet – Duplikatfilterung notwendig
  - Problem 2: Obermengen können seltene Teilmengen enthalten – Zählen überflüssig

Beispiel



Kandidatengenerierung: Besser

- generateCandidates: Finde alle Itemsets mit k Elementen, die nur häufige Teilmengen enthalten
  - Build downward: Berechne den (k-2)-Join: Alle k-Itemsets, die man aus häufigen (k-1)-Itemsets X, Y bilden kann mit $|X \cap Y|=k-2$ und $x_1=y_1, \ldots, x_{k-2}=y_{k-2}$
    - X und Y stimmen in den ersten k-2 Items überein und unterscheiden sich im (k-1)'ten
    - Benötigt nur (k-1)-Itemsets, effizient implementierbar
    - Löst Problem 1 – jeder Kandidat wird nur einmal aufgezählt
      - (Beweis folgt)
  - Prune upward: Für all diese potentiell häufigen Itemsets testen, ob alle ihre k-1 großen Teilmengen häufig (also in L_{k-1}) sind
    - Benötigt nur (k-1)-Itemsets, streicht alle seltenen Kandidaten
    - Löst Problem 2 – unnötiges Zählen sparen

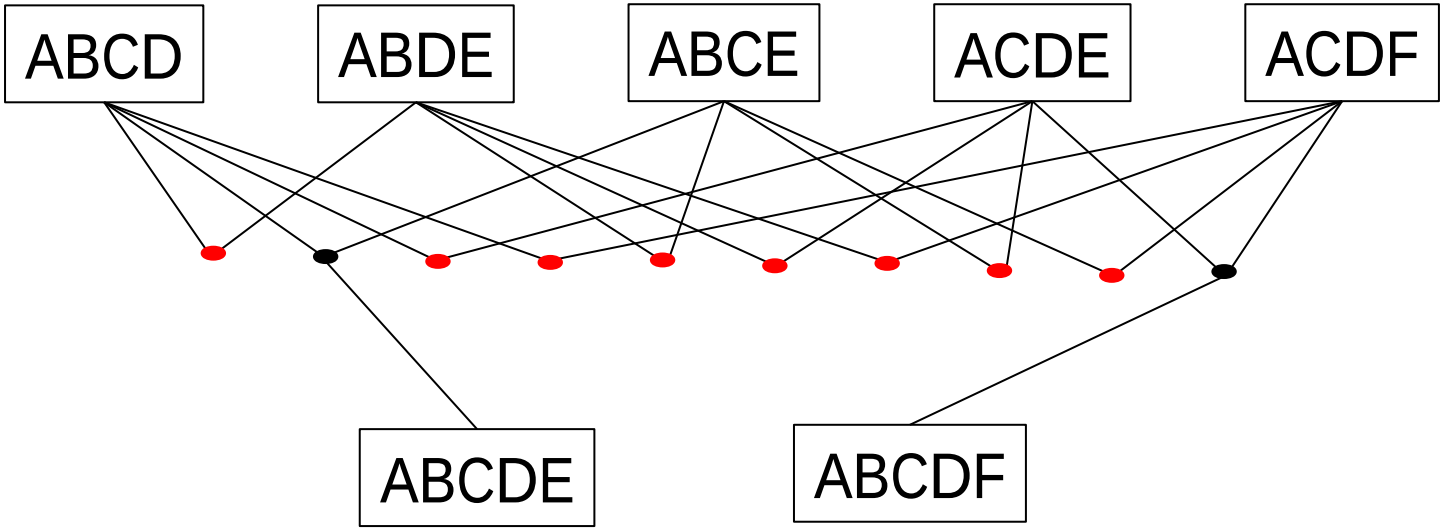
Warum der (k-2)-Join?

- Lemma
  Der (k-2)-Join findet alle potentiell häufigen k-Obermengen, die sich aus zwei häufigen (k-1)-Teilmengen zusammensetzen lassen, genau einmal.
- Beweis
  - „Findet alle k-Obermengen“: Sei l eine potentiell häufige Obermenge mit k Elementen, die sich aus zwei (k-1) Teilmengen zusammensetzen lässt und die der (k-2) Join nicht findet. Sortiere die Elemente von l={i1, i2, i3, i4, i5} und bilde die Teilmengen T1={i1, i2, i3, i4} und T2={i1, i2, i4, i5}.
    - Wenn T1 und T2 häufig sind und damit in L_{k-1}, wird der (k-2)-Join sie bilden – Widerspruch
    - Wenn T1 oder T2 nicht häufig ist, kann l nicht häufig sein, ist also keine potentiell häufige Obermenge - Widerspruch

Warum der (k-2)-Join?

- Lemma
  Der (k-2)-Join findet alle potentiell häufigen k-Obermengen, die sich aus zwei häufigen (k-1)-Teilmengen zusammensetzen lassen, genau einmal.
- Beweis
  - „... genau einmal“: Sei l eine potentiell häufige k-Obermenge, die sich aus zwei (k-1) Teilmengen T1 und T2 zusammensetzen lässt. Die ersten k-2 Elemente von l sind eindeutig bestimmt und müssen in T1 und T2 enthalten sein. Das vierte Element von l muss das vierte Element von T1 sein (oder T2), und das fünfte das vierte von T2 (oder T1). Damit sind T1 und T2 eindeutig bestimmt und l kann nur aus diesen beiden Teilmengen gebildet werden.

Beispiel



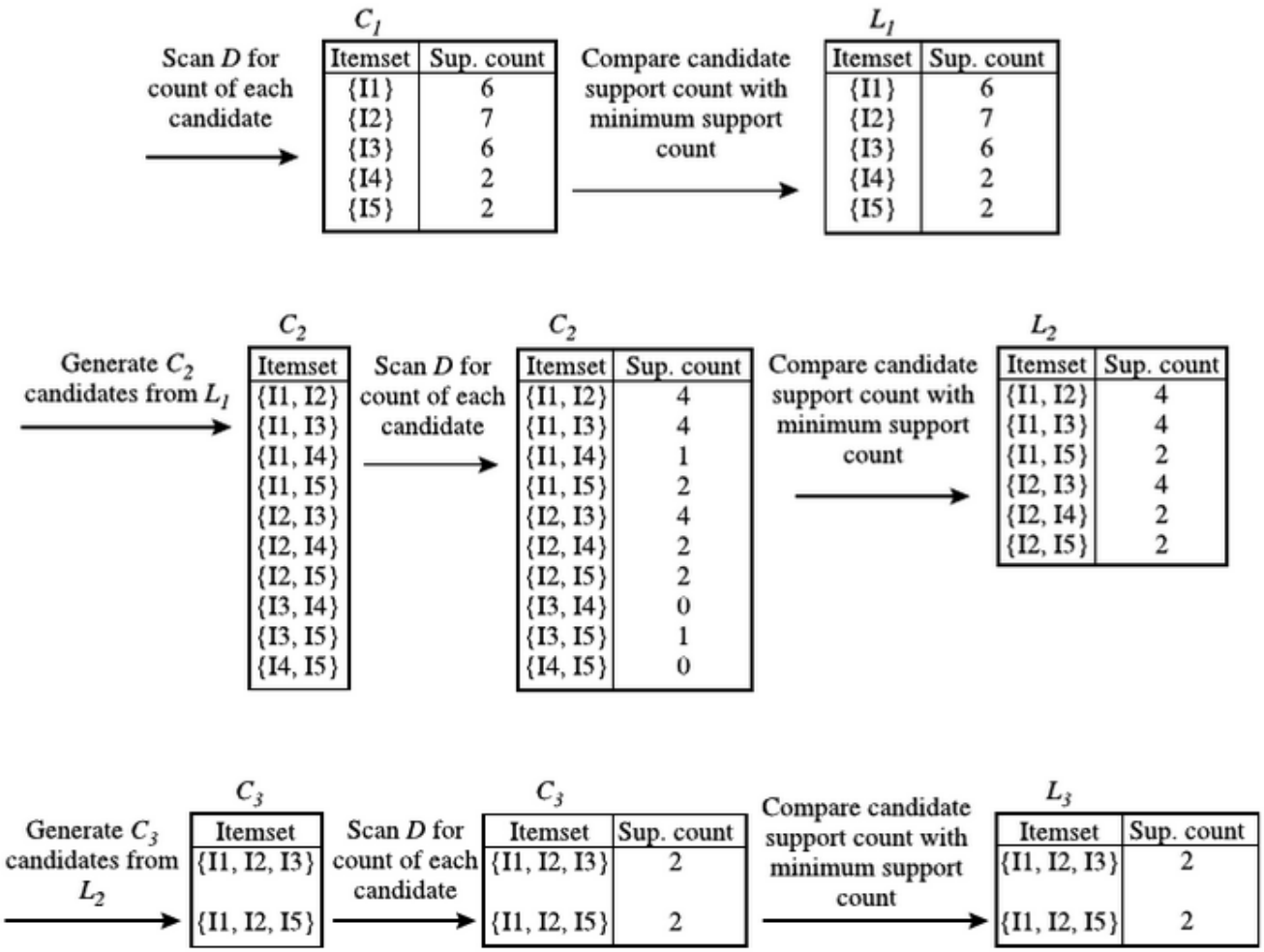
Was noch fehlt

- Warum nur die Obermengen von zwei Mengen aus l_{k-1}?
- Lemma
  Jede potentiell häufige k-Obermenge ist die Union von zwei häufigen (k-1)-Teilmengen.
- Beweis
  - Sei l eine potentiell häufige k-Obermenge, d.h., |l|=k. Wir ziehen aus l zufällig zwei Teilmengen T1, T2 mit je (k-1) Elementen. T1 und T2 müssen häufig sein, sonst ist l nicht potentiell häufig. Damit sind T1 und T2 in l_{k-1}. Ihre Union ergibt per Konstruktion l

Prune Upwards

- Der (k-2)-Join findet alle „bildbaren“ Kandidaten einmal
- Das heißt nicht, dass wir die alle zählen müssen
- Prune Upward: Prüfe für jeden Kandidaten l, ob alle seine (k-1)-Teilmengen häufig sind
  - Wir müssen alle k (k-1)-elementigen Teilmengen von l aufzählen und in L_{k-1} nachschlagen
  - Sehr oft k Teilmengen nachschlagen – teuer
- Verwendung spezieller Datenstrukturen
  - Effiziente Unterstützung der Operation subset(L_{k-1}, l_k)
    - Enthält die Menge L_{k-1} eine Teilmenge von l_k nicht?
  - Eine Möglichkeit: Hashbaum
  - Diverse weitere Strategien veröffentlicht

Beispiel (minsup=2)



Join	Upward pruning
I1-i2-i3	
I1-i3-i4	I3-i4 selten
I1-i2-i5	
I2-i3-i4	I3-i4 selten
I2-i3-i5	I3-i5 selten
I2-i4-i5	I4-i5 selten

Quelle: Han/Kamber, Data Mining

Inhalt dieser Vorlesung

- Association Rule Mining (ARM)
- A-Priori Algorithmus
  - Grundalgorithmus
  - Kandidatenerzeugung und –pruning
  - Datenstruktur für den Pruningschritt
  - Erweiterung: Partitionierung
  - Ableitung von Assoziationsregeln
- Varianten

Partitionierung [SON95]

- Auch das häufige Scannen der Datenbank ist teuer
- Möglichkeit: Partitionierung
  - Zerlege T in p nicht überlappende Partitionen
  - Wähle p so, dass jede Partition in den Hauptspeicher passt
  - Phase 1: Zähle alle häufigen Itemsets in jeder Partition
    - Sei der Support relativ zur Datenbankgröße angegeben (x%)
    - Es gilt: Itemset i kann nur häufig in T sein, wenn i häufig (in % der Partitionsgröße) in mindestens einer Partition ist
    - Suche in einem Scan alle häufigen Itemsets (L_p) in Partition p
  - Phase 2
    - Bilde Vereinigung L aller L_p
    - Scanne T und zähle Häufigkeit alle Mengen aus L

Bewertung

- Vorteile
  - Es werden nur zwei I/O-intensive Scans von T benötigt
  - Sehr schnell, wenn Anzahl Kandidaten klein (hoher minsup)
- Nachteil
  - Zweite Phase teuer bei vielen Kandidaten
  - Bei wirklich vielen Kandidaten kann man die Summen nicht mehr im Hauptspeicher halten

Finden der Assoziationsregeln

- Jedes häufige Itemset X kann |X| Assoziationsregeln „beinhalten“
- Welche davon genügen den Ansprüchen?
  - Alle Regel der Art $X \backslash x_i \rightarrow x_i$ bilden
  - Confidence berechnen
    - Da $X \backslash x_i$ und x_i häufig sein müssen, ist deren Support schon berechnet
- Kein neues Scannen der Datenbank notwendig

Inhalt dieser Vorlesung

- Einführung Association Rule Mining (ARM)
- A-Priori Algorithmus
- Varianten
  - Quantitative Regeln
  - ARM mit hierarchischen Itemsets
  - Interessante Regeln

Motivation

- Bisher haben wir nur binäre Items betrachtet
  - OK für Warenkorbanalyse
  - Nicht OK für Kundenanalyse
    - Was haben Kunden gemeinsam, die Versicherungsbetrug verüben?
    - Wir wissen Einkommen, durchschnittliche Ausgaben, Tarif, Versicherungsfälle der letzten Jahre, Wohnort, Familienstand, ...
    - Viele nicht-binären Items
- Quantitative Assoziationsregeln

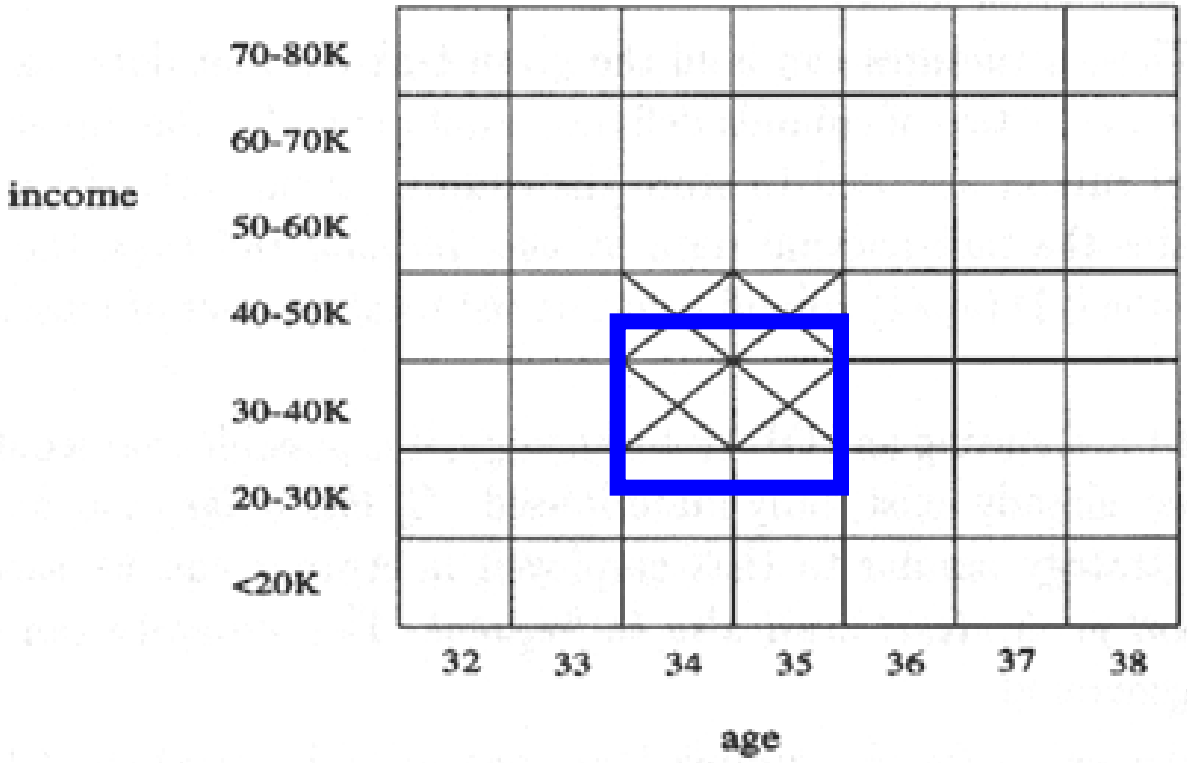
Quantitative Assoziationsregeln [SA96]

- Einfache Möglichkeit: Diskretisierung / Binning

ID	Alter:20..29	Alter:30..39	Alter:40..49
1	1	0	0
2	0	1	0

  - Vorteil: Einfach
  - Nachteil: Die Bins bestimmen, was man finden kann
    - Viele Bins: Viele Items, hohe Auflösung, aber Gefahr, an den Grenzen interessante Bereiche zu übersehen
    - Wenig Bins: Weniger Items, grobe Auflösung, geringere (aber immer noch vorhandene) Gefahr, etwas zu übersehen
- Besser: Dynamische Bestimmung der Bins aus den Daten während des ARM

Beispiel



- Echter Bereich: 33<X<35, 27500<Y<46000 → Gefahr
- Gefunden: 33<X<35, 30000<Y<50000 → Gefahr

Idee (ohne Details)

- Zunächst sehr feines Binning aller Attribute
- Zusammenfügen zu größeren zusammenhängenden Bereichen, die ausreichend Support haben
  - Ausreichend „grosse“ feine Bins bleiben auch erhalten
  - Führt zu überlappenden Bins
- Alle häufigen Bereiche werden als 1-Itemsets betrachtet und A-Priori gestartet
- Aber: Oftmals sehr viele Bereiche, viele Regeln, hohe Laufzeit

Beispiel (minsup=2)

ID	Alter	Fam.stand	# Autos
100	23	ledig	1
200	25	verheiratet	1
300	29	ledig	0
400	34	verheiratet	2
500	38	verheiratet	2

Intervall	Integer
20..24	1
25..29	2
30..34	3
35..39	4

ID	Alter	Fam.stand	# Autos
100	1	2	1
200	2	1	1
300	2	2	0
400	3	1	2
500	4	1	2

Itemset	Support
{20..24}	1
{25..29}	2
{30..34}	1
{34..39}	1
{20..29}	3
{30..39}	2
{verheiratet}	3
{ledig}	2
{30..39, verheiratet}	2
. . .	. . .

Hierarchische Itemsets [SA97]

- Dimensionen haben Klassifikationsstufen
- Wir suchen nicht Regeln der Art „SkischuhX1→SkiY1, SkischuhX2 → SkiY1“, sondern „Skischuhe→Ski“
- Items höher Klassifikationsstufen haben automatisch höheren Support
  - Dafür sind es weniger Items
  - Regeln auf unteren Ebenen haben geringeren Support und kommen eventuell nicht über minsup
  - Hoher Support: man findet abstraktere Regeln
- Vorsicht: Regeln können redundant sein
  - Immer, wenn „X→Y“ häufig ist, muss auch „vorfahr(X)→Y“ häufig sein
  - Nur die spezifischsten, aber noch häufigen Regeln finden

Interessante Regeln

- ARM Algorithmen produzieren oft riesige Regelmengen
- Regeln können irreführend sein
  - 60% der Schüler spielen Fußball, 75% essen Schokoriegel, 40% spielen Fußball und essen Schokoriegel
  - Assoziationsregeln
    - „Spielt Fußball“→ „Isst Schokoriegel“, Konfidenz = 67%
    - TRUE → „Isst Schokoriegel“, Konfidenz = 75%
  - „Fußball spielen“ und „Schokoriegel essen“ sind in Wahrheit negativ korreliert, die Regel hat trotzdem hohe Konfidenz
- Ursache: Zwei (oder k) häufige Items sind schon per Zufall auch in Kombination häufig
- Interessant sind die Assoziationen, die man nicht schon per Zufall erwarten würde

Interessantheits-Maß

- Sei rup(H) der relative Support eines Itemset H in T
  - rup(H) = sup(H)/|T|
- Sind Itemsets H_1, H_2 nicht korreliert, erwarten man

$$rup(H_1 \cup H_2) = rup(H_1) * rup(H_2)$$

- Ein einfaches Maß für die Abweichung von der zufällig erwarteten Häufigkeit ist dann (odds-score)

$$\text{int}(H_1, H_2) = \frac{rup(H_1 \cup H_2)}{rup(H_1) * rup(H_2)} \qquad \text{int}(H \rightarrow i) = \frac{rup(H \cup i)}{rup(H) * rup(i)}$$

  - int()=1: Häufigkeit ist per Zufall erwartet
  - int()<1: H und i werden auffallend selten zusammen gekauft
  - int()>1: H und i werden auffallend häufig zusammen gekauft

Literatur

- Agrawal, R. and Srikant, R. (1994). "Fast Algorithms for Mining Association Rules". 20th Int. Conf. Very Large Data Bases.
- Srikant, R. and Agrawal, R. (1996). "Mining quantitative association rules in large relational tables". SIGMOD Int. Conf. on Management of Data, New York, USA.
- Srikant, R. and Agrawal, R. (1997). "Mining generalized association rules." Future Generation Computer Systems 13(2-3): 161-180.
- Savasere, A., Omiecinski, E. and Navathe, S. (1995). "An Effective Algorithm for Mining Association Rules in Large Databases". 21st Int. Conf. on Very Large Databases, Zurich, Switzerland.
- Zhang, C. and Zhang, S. (2002). "Association Rule Mining - Models and Algorithms", Springer LNAI 2307.

Selbsttest

- Was besagt die A-Priori-Regel, die im Association Rule Mining verwendet wird?
- Definieren Sie Support und Konfidenz einer Assoziationsregel der Form $R=H \rightarrow i$
- Warum sind nicht alle Assoziationsregeln, die ARM findet, auch per-se interessant?
- Wie funktioniert der A-Priori-Algorithmus (Pseudo-Code angeben)?
- Zeigen Sie, dass der (k-2) Join alle potentiell häufigen Kandidaten aufzählt