



INTELLIGENTE DATENANALYSE IN MATLAB

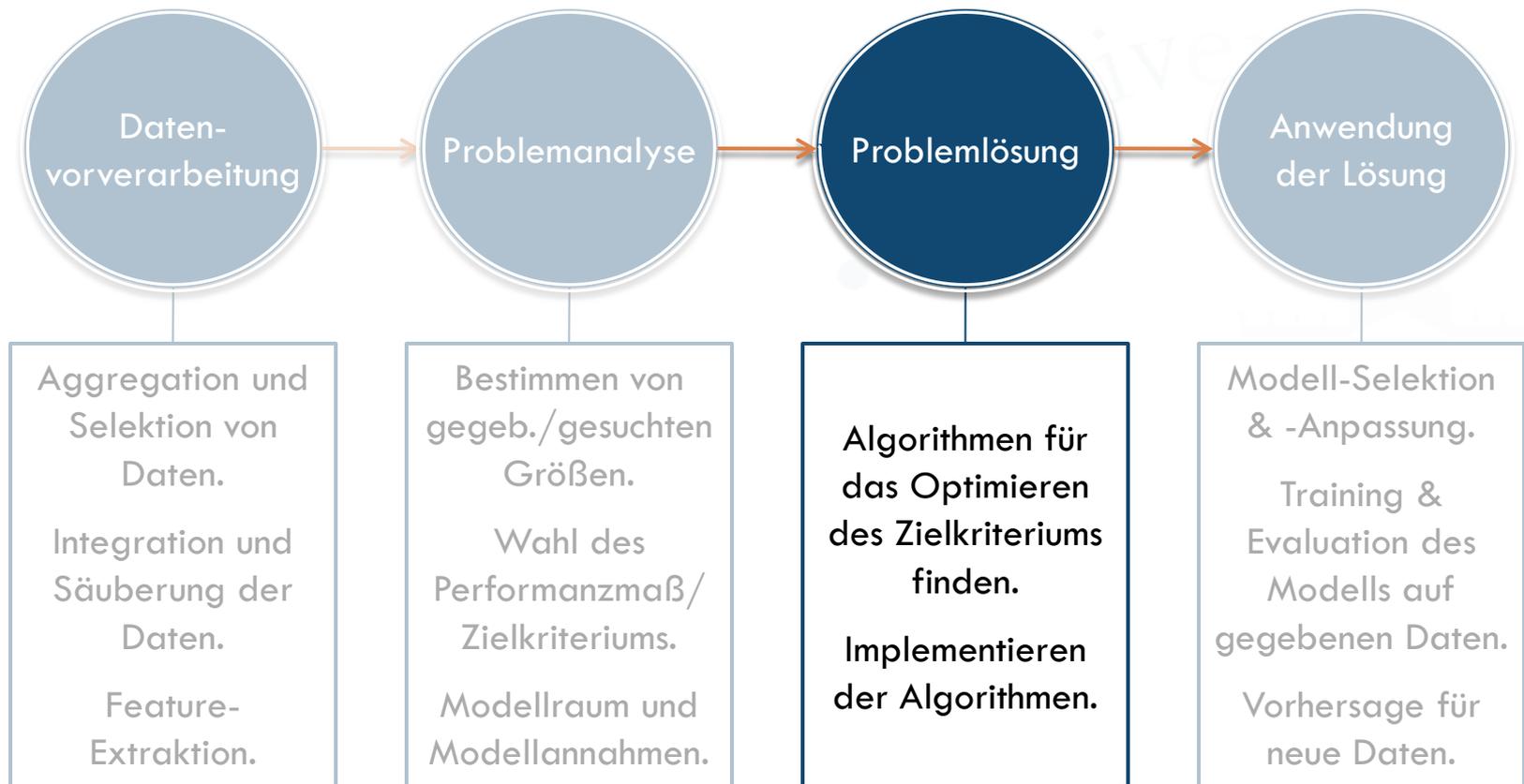
Unüberwachtes Lernen: Clustern von Attributen

Literatur

- J. Han, M. Kamber: Data Mining – Concepts and Techniques.
- J. Han et. al: Mining Frequent Patterns without Candidate Generation.
- C. Zhang, S. Zhang: Association Rule Mining.

Überblick

□ Schritte der Datenanalyse:



Unüberwachtes Lernen

Arten von Modellen & Lernproblemen

- Clustern von Instanzen:
 - Finden und deterministische bzw. probabilistische Zuweisung zu Bereichen mit vielen Datenpunkten (Cluster).
- Clustern von Attributen:
 - Häufig zusammen auftretende (ähnliche bzw. korrelierte) Attribute finden.
- Clustern von Instanzen & Attributen (Co-Clustern):
 - Gleichzeitiges Clustern von Instanzen und Attributen.
- Outlier Detection:
 - Suche nach seltenen/auffälligen Datenpunkten.

Clustern von Attributen

Problemstellung

- Gegeben: Menge von n Trainingsdaten $\mathbf{x}_i \in X^m$ mit m Attributen (Datenmatrix $\mathbf{X} \in X^{m \times n}$ mit Spaltenvektoren \mathbf{x}_i).
- Gesucht:
 - Numerische Attribute:
 - Cluster von ähnlichen, assoziierten oder redundanten Attributen analog zu Clustern von Instanzen (transponierte Datenmatrix segmentieren).
 - Diskrete Attribute:
 - Finden von *Frequent Itemsets*, d.h. häufig auftretender Muster bspw. für Kompression, Visualisierung etc.
 - Lernen von Assoziationsregeln der Form „wenn die Attribute 1, 2, 3 die Werte a, b, c annehmen, dann haben die Attribute 4, 5 die Werte d, e “.

Clustern von Attributen

Anwendung

- Warenkorbanalyse.
 - z.B. für Katalog-Design, Cross-/Up-Selling, Empfehlungssysteme, Produktanordnung in Supermärkten usw.
- Analyse des Surfverhaltens von Internetnutzern.
 - z.B. zur Optimierung der Navigation auf einer Webseite.
- Gruppierung von Dokumenten.
 - z.B. für Email-Batch-Erkennung, Plagiat-Erkennung.

Lernen von Assoziationsregeln

- Gesucht sind Assoziationsregeln der Form „wenn die Attribute p_i der Menge $P = \{p_1, \dots, p_s\} \subset \{1, \dots, m\}$ bestimmte Werte y_i annehmen, dann haben die Attribute q_j der Menge $Q = \{q_1, \dots, q_t\} \subset \{1, \dots, m\}$, $P \cap Q = \emptyset$ die Werte y'_j “.

- Definition: Boolesche Funktion

$$g(\mathbf{x}, p, y) = \begin{cases} \text{wahr} & x_p = y \\ \text{falsch} & x_p \neq y \end{cases}$$

- Definition: Assoziationsregel

$$g(\mathbf{x}, p_1, y_1) \wedge \dots \wedge g(\mathbf{x}, p_s, y_s) \Rightarrow g(\mathbf{x}, q_1, y'_1) \wedge \dots \wedge g(\mathbf{x}, q_t, y'_t)$$

Lernen von Assoziationsregeln

Beispiel

- Beispiel Warenkorbanalyse:
Welche Produkte werden (nicht)
zusammen gekauft?

Kunde	Brot	Milch	Eier	Bier	Müsli
1	nein	ja	ja	nein	ja
2	ja	ja	nein	ja	nein
3	nein	nein	ja	ja	nein
4	nein	ja	ja	nein	ja

- Mögliche boolesche Fragen:

- Kaufte Kunde 1 Milch?

$$g(\mathbf{x}_1, \text{Milch}, \text{ja}) = \text{wahr}$$

- Kaufte Kunde 3 keine Eier?

$$g(\mathbf{x}_3, \text{Eier}, \text{nein}) = \text{falsch}$$

- Mögliche Assoziationsregeln:

- Wer Milch kauft, kauft auch Müsli:

$$g(\mathbf{x}, \text{Milch}, \text{ja}) \Rightarrow g(\mathbf{x}, \text{Müsli}, \text{ja})$$

- Wer Bier und Eier kauft, kauft kein Brot:

$$g(\mathbf{x}, \text{Bier}, \text{ja}) \wedge g(\mathbf{x}, \text{Eier}, \text{ja}) \Rightarrow g(\mathbf{x}, \text{Brot}, \text{nein})$$

Lernen von Assoziationsregeln

Dateneigenschaften

- Wertebereich der Attribute:
 - Binäre/boolesche, nominale oder ordinale Attribute.
 - Numerische Attribute: müssen zuvor diskretisiert werden.
- Bedeutung der Attribute:
 - Verschiedene Attribute/Belegungen unterschiedlich „wichtig“.
 - z.B. „Wer Computer kauft, kauft auch Computer-Zubehör“ ist evtl. informativer als „Wer einen Dell-PC kauft, kauft auch eine Maus“.
 - z.B. „Wer Milch kauft, kauft auch Müsli“ ist evtl. informativer als „Wer Milch kauft, kauft kein Bier“.
 - Redundanz, d.h. Ähnlichkeit/Korrelation zwischen Attributen führt zu semantisch gleichen (redundanten) Regeln.

Lernen von Assoziationsregeln

Bewertung von Assoziationsregeln

- Es gibt exponentiell viele mögliche Assoziationsregeln.
- Assoziationsregeln unterschiedlich stark durch Daten gestützt, d.h. unterschiedlich wahrscheinlich.

□ Betrachten Ereignisse:

$$A = g(\mathbf{x}, p_1, y_1) \wedge \dots \wedge g(\mathbf{x}, p_s, y_s)$$

$$B = g(\mathbf{x}, q_1, y'_1) \wedge \dots \wedge g(\mathbf{x}, q_t, y'_t)$$

□ Frage: Wie wahrscheinlich ist eine Assoziationsregel $A \Rightarrow B$ gegeben die Daten?

$$P(A \Rightarrow B | \mathbf{X}) = P(\neg A \vee B | \mathbf{X}) = 1 - P(A | \mathbf{X}) + P(A, B | \mathbf{X})$$

□ Aus $P(A | \mathbf{X}) = 0$ folgt jedoch $P(A \Rightarrow B | \mathbf{X}) = 1$.

Lernen von Assoziationsregeln

Bewertung von Assoziationsregeln

□ Betrachten daher zwei Kriterien:

- Rückhalt (*support*), d.h. wie häufig treten die Ereignisse A und B gemeinsam ein:

$$\text{supp}(A \Rightarrow B) = P(A, B | \mathbf{X})$$

- Vertrauen (*confidence*), d.h. wie oft tritt B ein wenn A eingetreten ist:

$$\text{conf}(A \Rightarrow B) = P(B | A, \mathbf{X}) = P(A, B | \mathbf{X}) / P(A | \mathbf{X})$$

- Assoziationsregel $A \Rightarrow B$ ist *stark* mit Mindest-Support s_{\min} und Mindest-Konfidenz c_{\min} falls:

$$\text{supp}(A \Rightarrow B) \geq s_{\min} \quad \text{und} \quad \text{conf}(A \Rightarrow B) \geq c_{\min}$$

Lernen von Assoziationsregeln

Beispiel (Fortsetzung)

□ Beispiel für $s_{\min} = 40\%$ und $c_{\min} = 60\%$:

- Wer Milch kauft, kauft auch Müsli:

$$\text{supp} = P(g(\mathbf{x}, \text{Milch}, \text{ja}), g(\mathbf{x}, \text{Müsli}, \text{ja})) = \frac{2}{4}$$

$$\text{conf} = P(g(\mathbf{x}, \text{Müsli}, \text{ja}) \mid g(\mathbf{x}, \text{Milch}, \text{ja})) = \frac{2}{3}$$

Kunde	Brot	Milch	Eier	Bier	Müsli
1	nein	ja	ja	nein	ja
2	ja	ja	nein	ja	nein
3	nein	nein	ja	ja	nein
4	nein	ja	ja	nein	ja

⇒ Starke Assoziationsregel $g(\mathbf{x}, \text{Milch}, \text{ja}) \Rightarrow g(\mathbf{x}, \text{Müsli}, \text{ja})$

- Wer Bier und Eier kauft, kauft kein Brot:

$$\text{supp} = P(g(\mathbf{x}, \text{Bier}, \text{ja}), g(\mathbf{x}, \text{Eier}, \text{ja}), g(\mathbf{x}, \text{Brot}, \text{nein})) = \frac{1}{4}$$

$$\text{conf} = P(g(\mathbf{x}, \text{Brot}, \text{nein}) \mid g(\mathbf{x}, \text{Bier}, \text{ja}), g(\mathbf{x}, \text{Eier}, \text{ja})) = 1$$

⇒ Schwache Assoziationsregel $g(\mathbf{x}, \text{Bier}, \text{ja}) \wedge g(\mathbf{x}, \text{Eier}, \text{ja}) \Rightarrow g(\mathbf{x}, \text{Brot}, \text{nein})$

Lernen von Assoziationsregeln

Lösungsansatz

- *Frequent-Itemset-Suche*: Finde zunächst alle Mengen

$$I = \{g(\mathbf{x}, p_1, y_1), \dots, g(\mathbf{x}, p_h, y_h)\}$$

mit Mindest-Support, d.h. $P(g(\mathbf{x}, p_1, y_1), \dots, g(\mathbf{x}, p_h, y_h)) \geq s_{\min}$.

- *Regel-Extraktion*: Für jede dieser Mengen I , bilde alle nicht-leeren Teilmengen $D \subset I$ und prüfe ob

$$\text{conf}(D \Rightarrow I \setminus D) = \frac{P(I)}{P(D)} \geq c_{\min}$$

erfüllt ist; falls ja, ist $D \Rightarrow I \setminus D$ starke Assoziationsregel.

Frequent-Itemset-Suche

- Für eine gegebene Menge $I = \{g(\mathbf{x}, p_1, y_1), \dots, g(\mathbf{x}, p_h, y_h)\}$ gelten folgende Aussagen:
 - ▣ Der Support von I ist größer oder gleich dem Mindest-Support, d.h. $\text{supp}(I) = P(g(\mathbf{x}, p_1, y_1), \dots, g(\mathbf{x}, p_h, y_h)) \geq s_{\min}$.
 - ↔ Ereignisse $g(\mathbf{x}, p_1, y_1), \dots, g(\mathbf{x}, p_h, y_h)$ wurden in den Daten häufig gemeinsam beobachtet.
 - ↔ Menge I bildet ein *Frequent Itemset*.
 - ⇒ Jede Teilmenge von I bildet ein *Frequent Itemset*.

Frequent-Itemset-Suche

- A-Priori-Eigenschaft: Jede Teilmenge eines Frequent Itemsets ist ein Frequent Itemset.
 - Beispiel: Wenn Milch & Müsli oft gekauft wird, wird auch Milch oft gekauft bzw. wird auch Müsli oft gekauft.
 - Folge: Frequent Itemset mit $k + 1$ Elementen besteht aus Vereinigung zweier k -elementiger Frequent Itemsets (mit $k - 1$ gleichen Elementen).
 - Beginnend mit 1-elementigen Frequent Itemsets lassen sich so beliebig große Frequent Itemsets finden (Apriori-Algorithmus).

Frequent-Itemset-Suche

Apriori-Algorithmus

□ Algorithmus:

Apriori (Datenmatrix \mathbf{x} , Mindest-Support s_{\min})

Setze $k = 0, C_1 = \{\{g(\mathbf{x}, i, y_i)\} | 1 \leq i \leq m, y_i \in Y_i\}$

DO

$k = k + 1$

$L_k = \emptyset, C_{k+1} = \emptyset$

FOR $I \in C_k$

IF $\text{supp}(I) \geq s_{\min}$ THEN

FOR $J \in L_k$

IF $|I \cup J| = k + 1$ THEN

$C_{k+1} = C_{k+1} \cup \{I \cup J\}$

$L_k = L_k \cup \{I\}$

WHILE $C_{k+1} \neq \emptyset$

RETURN $L_1 \cup \dots \cup L_k$

C_k sind alle Kandidaten für Frequent Itemsets der Größe k

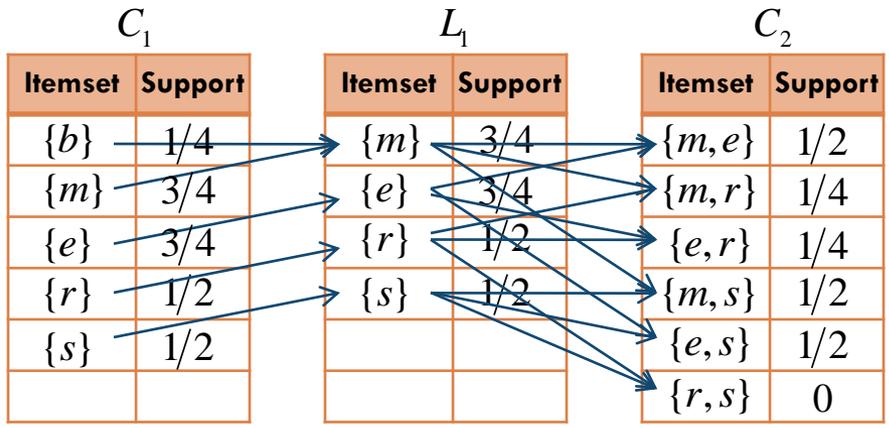
L_k sind alle Frequent Itemsets der Größe k

I und J sind Frequent Itemsets der Größe $k - 1$ gleichen Elementen

Frequent-Itemset-Suche

Apriori-Algorithmus: Beispiel

- Betrachten beispielhaft nur „positive“ Items und $s_{\min} = 50\%$:



$$L_1 = \{\{m\}, \{e\}, \{r\}, \{s\}\}$$

Kunde	Brot	Milch	Eier	Bier	Müsli
1	nein	ja	ja	nein	ja
2	ja	ja	nein	ja	nein
3	nein	nein	ja	ja	nein
4	nein	ja	ja	nein	ja

Elementarereignisse (Items)

- $b = g(\mathbf{x}, \text{Brot}, \text{ja})$
- $\bar{b} = g(\mathbf{x}, \text{Brot}, \text{nein})$
- $m = g(\mathbf{x}, \text{Milch}, \text{ja})$
- $\bar{m} = g(\mathbf{x}, \text{Milch}, \text{nein})$
- $e = g(\mathbf{x}, \text{Eier}, \text{ja})$
- $\bar{e} = g(\mathbf{x}, \text{Eier}, \text{nein})$
- $r = g(\mathbf{x}, \text{Bier}, \text{ja})$
- $\bar{r} = g(\mathbf{x}, \text{Bier}, \text{nein})$
- $s = g(\mathbf{x}, \text{Müsli}, \text{ja})$
- $\bar{s} = g(\mathbf{x}, \text{Müsli}, \text{nein})$

Frequent-Itemset-Suche

Apriori-Algorithmus: Beispiel

- Betrachten beispielhaft nur „positive“ Items und $s_{\min} = 50\%$:

Itemset	Support
{m, e}	1/2
{m, r}	1/4
{e, r}	1/4
{m, s}	1/2
{e, s}	1/2
{r, s}	0

Itemset	Support
{m, e}	1/2
{m, s}	1/2
{e, s}	1/2

Itemset	Support
{m, e, s}	1/2

$$L_1 = \{\{m\}, \{e\}, \{r\}, \{s\}\}$$

$$L_2 = \{\{m, e\}, \{m, s\}, \{e, s\}\}$$

Kunde	Brot	Milch	Eier	Bier	Müsli
1	nein	ja	ja	nein	ja
2	ja	ja	nein	ja	nein
3	nein	nein	ja	ja	nein
4	nein	ja	ja	nein	ja

Elementarereignisse (Items)

$$\left\{ \begin{array}{l}
 b = g(\mathbf{x}, \text{Brot}, \text{ja}) \\
 \bar{b} = g(\mathbf{x}, \text{Brot}, \text{nein}) \\
 m = g(\mathbf{x}, \text{Milch}, \text{ja}) \\
 \bar{m} = g(\mathbf{x}, \text{Milch}, \text{nein}) \\
 e = g(\mathbf{x}, \text{Eier}, \text{ja}) \\
 \bar{e} = g(\mathbf{x}, \text{Eier}, \text{nein}) \\
 r = g(\mathbf{x}, \text{Bier}, \text{ja}) \\
 \bar{r} = g(\mathbf{x}, \text{Bier}, \text{nein}) \\
 s = g(\mathbf{x}, \text{Müsli}, \text{ja}) \\
 \bar{s} = g(\mathbf{x}, \text{Müsli}, \text{nein})
 \end{array} \right.$$

Frequent-Itemset-Suche

Apriori-Algorithmus: Beispiel

□ Betrachten beispielhaft nur „positive“ Items und $s_{\min} = 50\%$:

Itemset	Support
$\{m, e, s\}$	$1/2$

Itemset	Support
$\{m, e, s\}$	$1/2$

Itemset	Support

- $L_1 = \{\{m\}, \{e\}, \{r\}, \{s\}\}$
- $L_2 = \{\{m, e\}, \{m, s\}, \{e, s\}\}$
- $L_3 = \{\{m, e, s\}\}$

Kunde	Brot	Milch	Eier	Bier	Müsli
1	nein	ja	ja	nein	ja
2	ja	ja	nein	ja	nein
3	nein	nein	ja	ja	nein
4	nein	ja	ja	nein	ja

- Elementarereignisse (Items)
- $b = g(\mathbf{x}, \text{Brot}, \text{ja})$
 - $\bar{b} = g(\mathbf{x}, \text{Brot}, \text{nein})$
 - $m = g(\mathbf{x}, \text{Milch}, \text{ja})$
 - $\bar{m} = g(\mathbf{x}, \text{Milch}, \text{nein})$
 - $e = g(\mathbf{x}, \text{Eier}, \text{ja})$
 - $\bar{e} = g(\mathbf{x}, \text{Eier}, \text{nein})$
 - $r = g(\mathbf{x}, \text{Bier}, \text{ja})$
 - $\bar{r} = g(\mathbf{x}, \text{Bier}, \text{nein})$
 - $s = g(\mathbf{x}, \text{Müsli}, \text{ja})$
 - $\bar{s} = g(\mathbf{x}, \text{Müsli}, \text{nein})$

Frequent-Itemset-Suche

Apriori-Algorithmus: Implementierung

- Mengenoperationen:
 - Verwendung von Hashs bzw. Hash-Bäumen.
- Berechnung des Supports:
 - Ignoriere Items die nicht Element eines Frequent Itemset der Größe k sind bei Berechnungen für Itemsets der Größe $>k$.
- Partitionierung der Daten:
 - Frequent Itemset bzgl. aller Daten muss Frequent Itemset mindestens einer Partition sein.
 - Berechne Frequent Itemsets aller Partitionen und prüfe welche davon auch Frequent Itemsets bzgl. aller Daten sind.

Frequent-Itemset-Suche

Apriori-Algorithmus: Bewertung

- Vorteile:
 - Effektiv und relativ einfach zu implementieren.
- Nachteile:
 - Sehr viele Kandidaten (hohe Laufzeit/Speicherverbrauch).
 - 10^4 elementare Frequent Itemsets erzeugen 10^7 zwei-elementige Kandidaten.
 - Suche nach Frequent Itemset der Größe 100 erfordert mindestens $2^{100} \approx 10^{30}$ Kandidaten.
 - Viele Iterationen über die Daten.
 - Frequent Itemset der Größe k erfordert $k + 1$ Iterationen.

Frequent-Itemset-Suche

FP-Growth-Algorithmus

- Idee: Daten als Frequent-Pattern-Baum darstellen und daraus Frequent Itemsets extrahieren.
- Aufbau des FP-Baums:
 - Speichert nur Informationen welche zum Finden von Frequent Itemsets notwendig sind (*Sufficient Statistics*).
 - Benötigt lediglich 2 Iterationen über die Daten.
- Extraktion der Frequent Itemsets:
 - Divide & Conquer-Ansatz, d.h. Reduktion des Suchproblems in viele kleine Optimierungsprobleme.
 - Vermeidung aufwendiger Kandidaten-Generierung.

Frequent-Itemset-Suche

FP-Growth-Algorithmus: Aufbau des FP-Baums

- Bestimmung aller Elementarereignisse (Items) und Sortierung nach ihrer Häufigkeit.

Kunde	Brot	Milch	Eier	Bier	Müsli
1	nein	ja	ja	nein	ja
2	ja	ja	nein	ja	nein
3	nein	nein	ja	ja	nein
4	nein	ja	ja	nein	ja

Kunde	\bar{b}	m	e	r	\bar{r}	s	\bar{s}	b	\bar{m}	\bar{e}
1	1	1	1	0	1	1	0	0	0	0
2	0	1	0	1	0	0	1	1	0	1
3	1	0	1	1	0	0	1	0	1	0
4	1	1	1	0	1	1	0	0	0	0
Häufigkeit	3	3	3	2	2	2	2	1	1	1

- Entferne alle Attribute mit zu kleinem Support.

Elementarereignisse (Items)

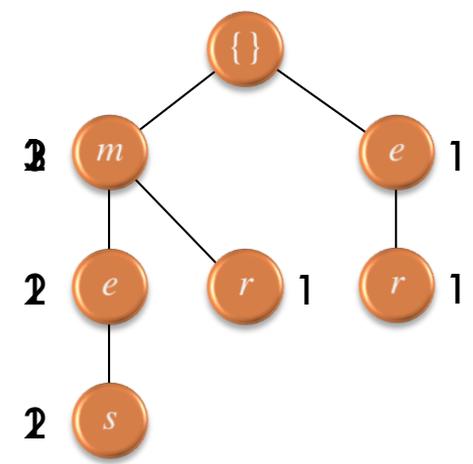
$$\begin{cases}
 b = g(\mathbf{x}, \text{Brot}, \text{ja}) \\
 \bar{b} = g(\mathbf{x}, \text{Brot}, \text{nein}) \\
 m = g(\mathbf{x}, \text{Milch}, \text{ja}) \\
 \bar{m} = g(\mathbf{x}, \text{Milch}, \text{nein}) \\
 e = g(\mathbf{x}, \text{Eier}, \text{ja}) \\
 \bar{e} = g(\mathbf{x}, \text{Eier}, \text{nein}) \\
 r = g(\mathbf{x}, \text{Bier}, \text{ja}) \\
 \bar{r} = g(\mathbf{x}, \text{Bier}, \text{nein}) \\
 s = g(\mathbf{x}, \text{Müsli}, \text{ja}) \\
 \bar{s} = g(\mathbf{x}, \text{Müsli}, \text{nein})
 \end{cases}$$

Frequent-Itemset-Suche

FP-Growth-Algorithmus: Aufbau des FP-Baums

- Betrachten beispielhaft nur „positive“ Items.
- Konstruktion des Präfix-Baums der Instanzen:

Kunde	<i>m</i>	<i>e</i>	<i>r</i>	<i>s</i>	<i>b</i>
1	1	1	0	1	0
2	1	0	1	0	1
3	0	1	1	0	0
4	1	1	0	1	0

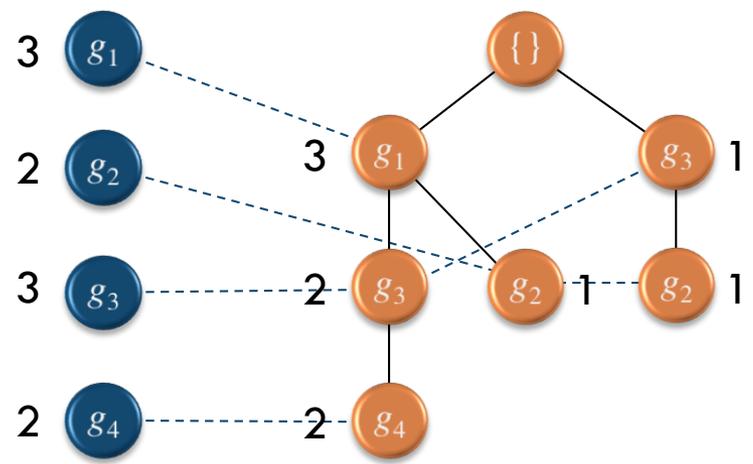


Nicht berücksichtigt bei Konstruktion des FP-Baums da zu kleiner Support

Frequent-Itemset-Suche

FP-Growth-Algorithmus: Extraktion der Frequent Itemsets

- Alle Frequent Itemsets, welche g_k enthalten und g_{k+1}, \dots, g_m nicht enthalten, sind Teilmengen eines Pfades von der Wurzel zu einem Knoten g_k .
- Idee: Rekursiv alle Frequent Itemsets extrahieren.



Frequent-Itemset-Suche

FP-Growth-Algorithmus: Extraktion der Frequent Itemsets

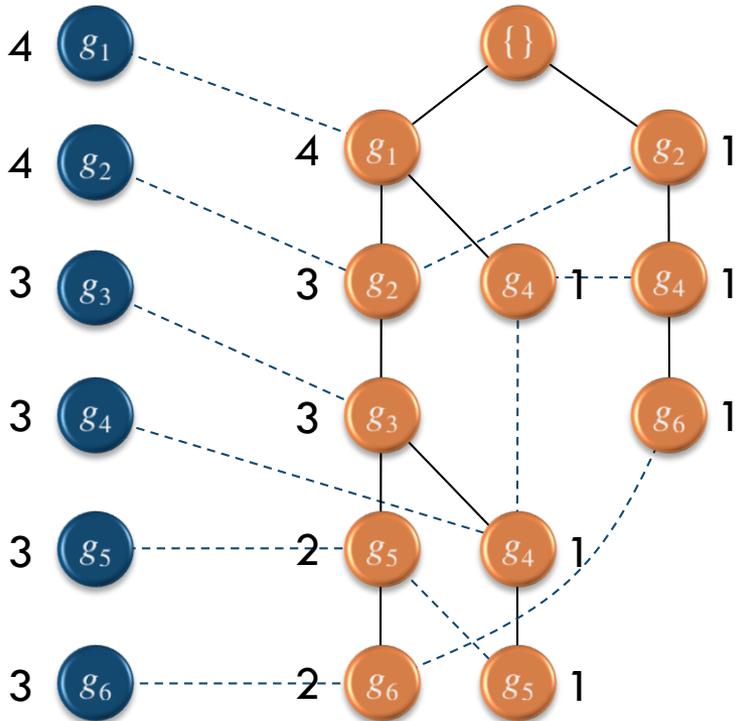
□ Algorithmus:

- Für alle Knoten (Items) g_k , betrachte Teilbaum mit allen Pfaden von der Wurzel zu einem Knoten (Item) g_k .
- Berechne Support der Items für diesen Teilbaum.
- Falls g_k ein Frequent Item ist:
 - Erzeuge FP-Baum mit allen Frequent Items g_p ($p < k$) des Teilbaumes und bestimme davon rekursiv alle Frequent Itemsets I_1, \dots, I_z .
 - Rückgabe aller Frequent Itemsets $\{g_k\}, I_1 \cup \{g_k\}, \dots, I_z \cup \{g_k\}$.
- Sonst, Rückgabe der leeren Menge.

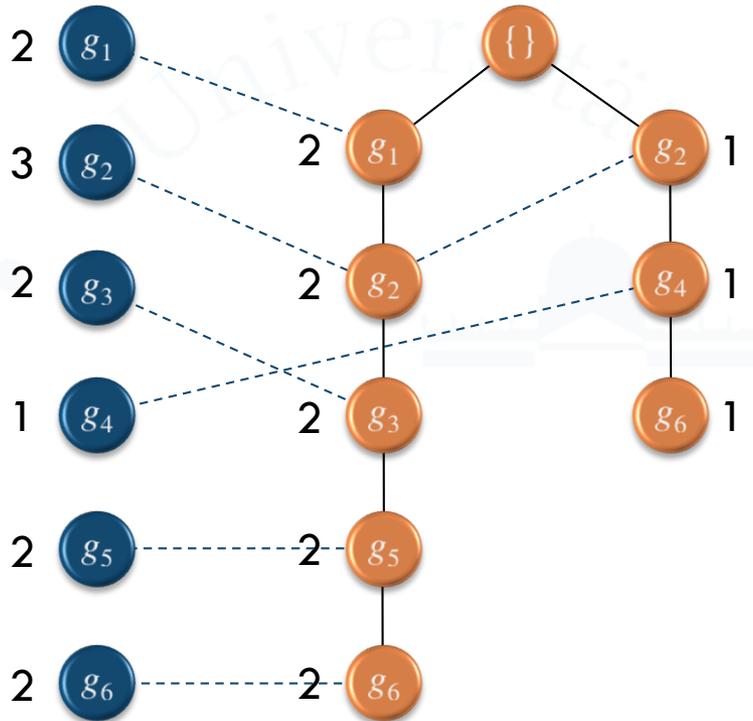
Frequent-Itemset-Suche

FP-Growth-Algorithmus: Extraktion der Frequent Itemsets

□ Beispiel für $n = 5, s_{\min} = 40\%$:



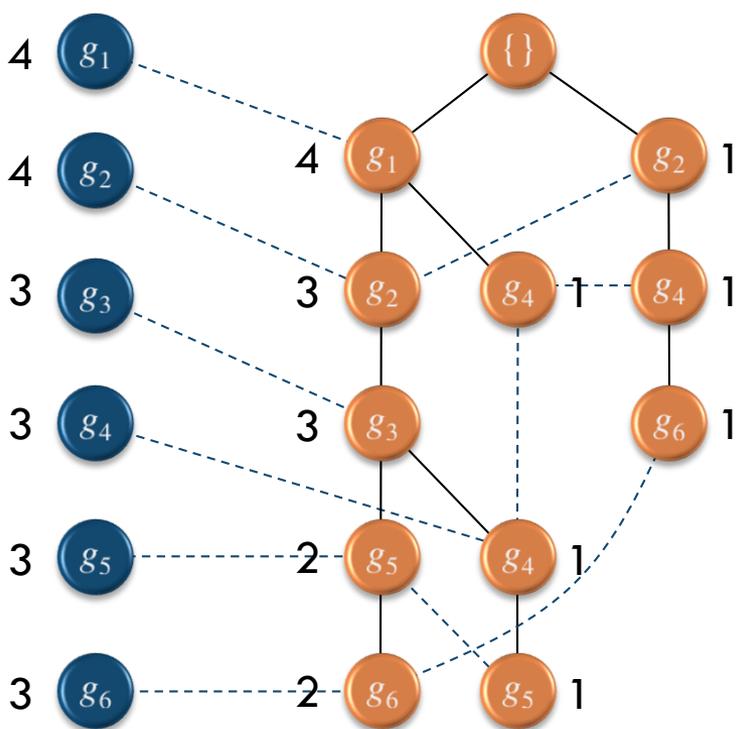
Teilbaum für g_6 :



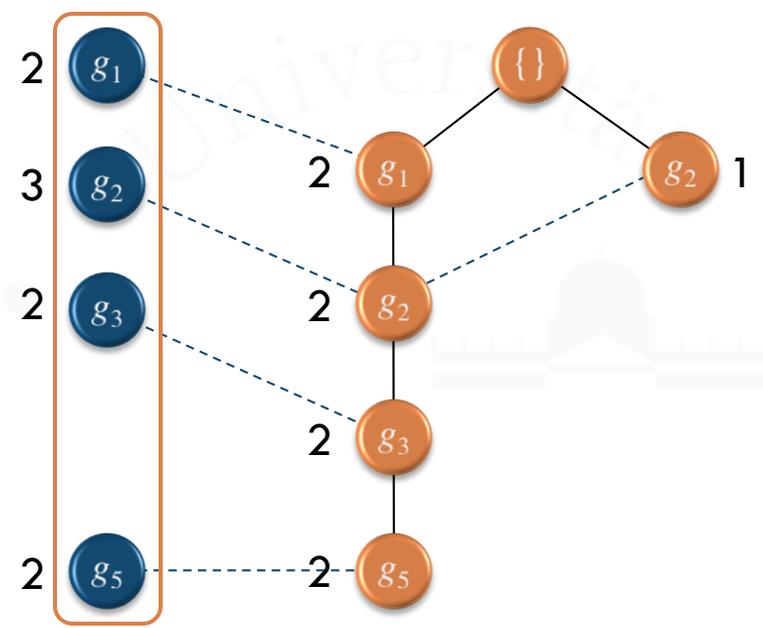
Frequent-Itemset-Suche

FP-Growth-Algorithmus: Extraktion der Frequent Itemsets

□ Beispiel für $n = 5, s_{\min} = 40\%$:



Teilbaum für g_6 :

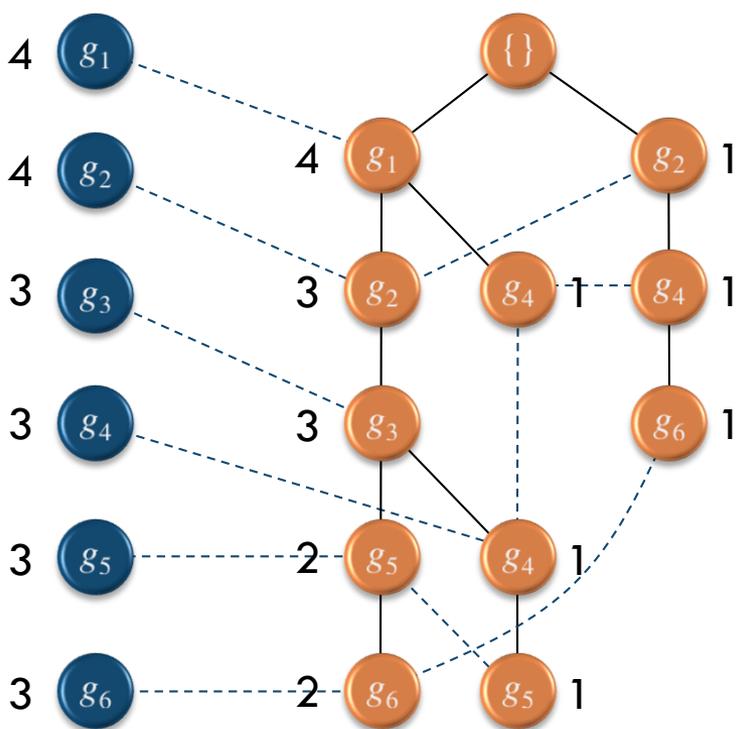


Frequent Items des Teilbaumes

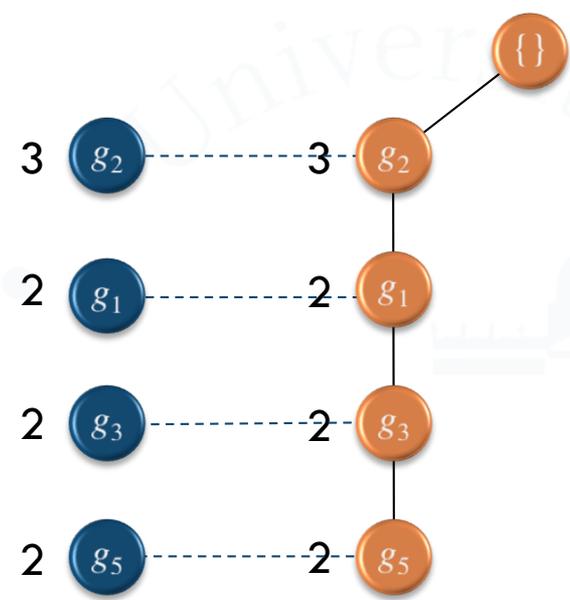
Frequent-Itemset-Suche

FP-Growth-Algorithmus: Extraktion der Frequent Itemsets

□ Beispiel für $n = 5, s_{\min} = 40\%$:



Teilbaum für g_6 :



FP-Baum des Teilbaumes

Frequent-Itemset-Suche

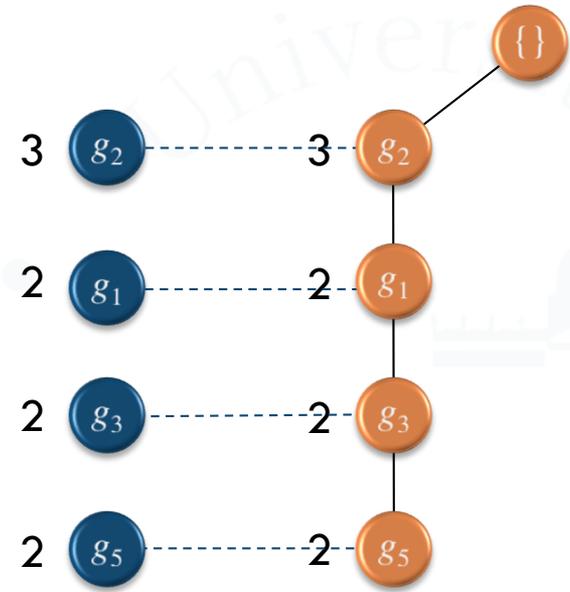
FP-Growth-Algorithmus: Extraktion der Frequent Itemsets

□ Beispiel für $n = 5, s_{\min} = 40\%$:

Teilbaum für g_6 :

Rekursion bzgl. g_6 ergibt:

- $\{g_2\}$
- $\{g_1\}, \{g_2, g_1\}$
- $\{g_3\}, \{g_2, g_3\}, \{g_1, g_3\}, \{g_2, g_1, g_3\}$
- $\{g_5\}, \{g_2, g_5\}, \{g_1, g_5\}, \{g_2, g_1, g_5\}, \{g_3, g_5\}, \{g_2, g_3, g_5\}, \{g_1, g_3, g_5\}, \{g_2, g_1, g_3, g_5\}$
- $\{g_6\}, \{g_2, g_6\}, \{g_1, g_6\}, \{g_2, g_1, g_6\}, \{g_3, g_6\}, \{g_2, g_3, g_6\}, \{g_1, g_3, g_6\}, \{g_2, g_1, g_3, g_6\}, \dots$



Frequent-Itemset-Suche

FP-Growth-Algorithmus: Bewertung

□ Vorteile:

- FP-Growth ist deutlich schneller als Apriori.
- Keine Kandidaten-Generierung.
- Iterationen über kompakte Datenstruktur (FP-Baum) statt über alle Daten.

□ Nachteile:

- Höherer Implementierungsaufwand.
- Innerhalb der Rekursion werden viele gleiche Bäume erzeugt; viele redundante Berechnungen.

Zusammenfassung

- Ziel: Identifikation ähnlicher bzw. korrelierter Attribute.
 - Assoziationsregeln, Frequent Itemsets, Sequenzmuster.
- Frequent-Itemset-Suche mit Kandidaten-Generierung (z.B. Apriori-Algorithmus):
 - Idee: Teilmenge eines Frequent Itemsets ist wieder ein Frequent Itemset \Rightarrow Konstruktion großer Itemsets aus kleinen.
- Frequent-Itemset-Suche ohne Kandidaten-Generierung (z.B. FP-Growth-Algorithmus):
 - Idee: Konstruktion eines Präfix-Baumes \Rightarrow Finden von Frequent Itemsets durch rekursives Traversieren und Restrukturieren des Baumes.