



Assoziationsregeln

Tobias Scheffer
Steffen Bickel

Assoziationsregeln

- Suche nach Regeln und Zusammenhängen in Datenbanken.
- Z.B. Analyse von „Basket Data“ in Supermärkten: Produkte, die zusammen gekauft wurden.
- Motivation
 - ◆ Cross-Marketing, Layout design
 - ◆ Loss-leader analysis
 - ◆ Kundensegmentierung.

Assoziationsregeln: Beispiel

- I = Menge von Items (z.B. {Chips, Bier, Windeln, Caçasa, Limetten, brauner_Zucker, Milch, ...})
- T = Menge von Transaktionen (z.B. { {Chips, Bier}, {Caçasa, Limetten, brauner_Zucker}, {Milch, Bier, Windeln}, ...})
- Beispiel-Assoziationsregeln
 - ◆ „Caçasa, Limetten \Rightarrow brauner_Zucker“
 - ◆ „Bier, Chips \Rightarrow Fernsehzeitschrift“
 - ◆ „Cornflakes, Babybrei \Rightarrow Milch, Windeln“

3

Assoziationsregeln: Problemstellung

- Gegeben
 - ◆ Menge I von Items,
 - ◆ Menge T von Transaktionen; $t \in T \Rightarrow t \subseteq I$.
- Finde alle Regeln $r = (X \Rightarrow Y)$ mit $X \subseteq I$, $Y \subseteq I$, so dass
 - ◆ Support $s(r) = \frac{|\{t \in T : X \cup Y \subseteq t\}|}{|\{t \in T\}|} \geq s_{\min}$
 - ◆ Confidence $c(r) = \frac{|\{t \in T : X \cup Y \subseteq t\}|}{|\{t \in T : X \subseteq t\}|} \geq c_{\min}$
- Support: „Allgemeinheit“ der Regel
- Confidence: „Genauigkeit“ der Regel

4

Assoziationsregeln: Apriori-Algorithmus

1. Finde häufige Itemsets; Itemsets mit Support $\geq s_{\min}$.
 1. Starte von Itemsets der Größe 1
 2. Erzeuge aus den Itemsets der Größe i die der Größe $i+1$.
2. Für alle gefundenen häufigen Itemsets, bilde alle Regeln über diesen Items und teste die Confidence.
3. Liefere alle Regeln mit ausreichend hohem Support und Confidence.

5

Suche nach häufigen Itemsets

- Häufige-Itemsets (s_{\min})
 1. $C_1 = \{ \{ i \} \mid i \in I \}$ (Itemsets der Größe 1)
 2. $L_1 = \text{Prune}(C_1)$.
 3. Solange $L_k \neq \{ \}$
 1. $C_{k+1} = \text{Generate}(L_k)$
 2. $L_{k+1} = \text{Prune}(C_{k+1}, s_{\min})$
 3. Increment k .
- Generate (L_k) [Version 1]
 1. Return $\{ \{ L_k \} \cup i \mid i \in I \}$

6

$L_k = \text{Prune}(C_k, s_{\min})$

1. Für alle $c \in C_k$: $\text{count}(c) = 0$.
 2. Für alle $t \in T$: (Datenbank-Scan)
 1. Für alle $c \subseteq t$: $\text{increment count}(c)$.
 3. Return $\{c \in C_k \mid \text{count}(c) \geq s_{\min}\}$
- Finden wir so wirklich alle häufigen Itemsets? Das Pruning beschränkt den Suchraum.

7

Suche nach häufigen Itemsets

- Wie hoch kann der Support von $(X \Rightarrow Y)$ sein, wenn $s(X \cup Y) = \text{sup}$ ist?
- Kann der Support $s(X \cup Y)$ größer sein als der Support $s(X)$?
- Für alle $c \in \text{Generate}(X)$: $s(c) \leq s(X)$.
- Suchraum nach Itemsets ist baumförmig; der Support sinkt in jedem Ast monoton.

8

Bilden von Regeln aus häufigen Itemsets

- Für alle gefundenen häufigen Itemsets, bilde alle Regeln über diesen Items und teste die Confidence.

- $$c(X \Rightarrow Y) = \frac{|\{t \in T : X \cup Y \subseteq t\}|}{|\{t \in T : X \subseteq t\}|} = \frac{s(X \cup Y)}{s(X)}$$

- Support haben wir schon für alle häufigen Itemsets berechnet.

9

Bilden von Regeln über Itemset

- Finde alle Regeln über die häufigen Itemsets I_k , mit $k \geq 2$ Elementen

1. Für alle $X \in I_k$

1. Für alle $Y \subseteq X$, Y nicht leer.

1. Wenn $c(X \setminus Y \Rightarrow Y) = \frac{s(X)}{s(X \setminus Y)} \geq c_{\min}$
 2. Dann gib Regel $(X \setminus \{Y\} \Rightarrow Y)$ aus.

10

Apriori: Beispiel

	Gin	Tonic	Kaluha	Chips	Cacasa	Limetten	Zucker
1		1	1	1			
2	1	1		1			
3					1	1	1
4					1	1	1
5	1	1	1				
6				1	1	1	1
7	1	1			1	1	1

- $s_{\min} = 3$
- $c_{\min} = 1$

11

Bilden von Regeln aus häufigen Itemsets (Optimierung)

- Pruning-Strategie für die Bildung von Regeln

- Beobachtung

$$c(XY \Rightarrow Z) = \frac{|\{t \in T : X \cup Y \cup Z \in t\}|}{|\{t \in T : X \cup Y \in t\}|}$$

$$\geq \frac{|\{t \in T : X \cup Y \cup Z \in t\}|}{|\{t \in T : X \in t\}|} = c(X \Rightarrow YZ)$$

- Wenn also schon $(XY \Rightarrow Z)$ unter c_{\min} liegt, dann müssen wir $(X \Rightarrow YZ)$ und $(Y \Rightarrow XZ)$ nicht mehr testen.

12

Bilden von Regeln über Itemset

- Finde alle Regeln über die häufigen Itemsets I_k , mit $k \geq 2$ Elementen
- 1. $m=1$.
- 2. $H_m =$ alle einelementigen Teilmengen von I_k .
- 3. Für $m = 1 \dots k-1$
 1. Für alle $h \in H_m$, $r = (I_k \setminus \{h\} \Rightarrow h)$
 - Wenn $c(I_k \setminus \{h\} \Rightarrow h) = \frac{s(I_k)}{s(I_k \setminus \{h\})} \geq c_{\min}$
 - Dann gib Regel r aus.
 - Sonst $H_m = H_m \setminus \{h\}$
 2. $H_{m+1} =$ Generate (H_m)

13

Generate (L_k) (Optimierung)

- Alle Teilmengen eines häufigen Itemsets sind auch wieder häufige Itemsets.
- Idee: Generierung häufiger Itemsets $m+1$ durch Kombination häufiger Itemsets der Größe m .
- Generate (L_k)
 1. $C_{k+1} = \{ \}$
 2. Für alle $X \in L_k$, Für alle $Y \in L_k$
 1. Wenn sich X und Y nur im letzten Element unterscheiden und das letzte Element von X kleiner als das von Y ist, dann $C_{k+1} = C_{k+1} + X \cup Y$

14

Generate (L_k) (Optimierung)

- Korrektheit? Vollständigkeit?
- Supermarkt-Beispiel mit neuem Generate-Algorithmus und neuem Regel-Generator-Algorithmus

15

Apriori TID

- Vor allem Datenbankzugriffe verursachen Kosten bei Apriori.
- Für großes k gibt es viele Datenbankeinträge, die zu keinem häufigen Itemset gehören, trotzdem wird auf diese zugegriffen.
- Apriori TID: Für jeden Eintrag wird die Liste aller häufigen Itemsets gespeichert, zu denen der Eintrag gehört. Wenn die Liste leer ist, wird der Eintrag gelöscht.

16

Apriori TID

- Häufige-Itemsets (s_{\min})
- 1. $C_1 = \{\{i\} \mid i \in I\}$ (Itemsets der Größe 1)
- 2. $L_1 = \text{Prune}(C_1)$; $k=1$; $\hat{C}_1 = \text{Datenbank}$ (einelementige TM)
- 3. Solange $L_k \neq \{\}$
 1. $C_{k+1} = \text{Generate}(L_k)$
 2. $\hat{C}_{k+1} = \{\}$
 3. Für alle Einträge $t \in \hat{C}_k$:
 1. $C_t = \{c \in C_{k+1} \mid (c \setminus c[k] \in t.\text{Menge_der_Itemsets} \text{ und } c[k+1] \in t.\text{Menge_der_Itemsets})\}$
 2. Für alle $c \in C_t$ Inkrementiere $c.\text{count}$.
 3. Wenn $C_t \neq \{\}$ dann $\hat{C}_{k+1} = \hat{C}_{k+1} \cup \langle t.\text{TID}, C_t \rangle$
 4. $L_{k+1} = \text{Prune}(C_{k+1}, s_{\min})$
 5. Inkrementiere k .

17

Apriori TID: Beispiel

- | | |
|--------------|-------------------------------------------------------------------------------------------------------------|
| ■ D: | ■ $\hat{C}_1 = \langle 0.\{\{1\},\{2\},\{3\},\{4\}\rangle, \dots, \langle 7,\{\{4\},\{5\}\}\rangle \rangle$ |
| ■ 0: 1 2 3 4 | ■ $C_1 = \{\{1\}, \dots, \{9\}\}$ |
| ■ 1: 1 2 6 | ■ $L_1 = ?$ |
| ■ 2: 1 2 3 5 | ■ $C_{k+1} = ?$ |
| ■ 3: 1 2 3 8 | ■ $\hat{C}_2 = ?$ |
| ■ 4: 1 3 9 | ■ ... |
| ■ 5: 2 3 9 | |
| ■ 6: 3 7 8 | |
| ■ 7: 4 5 | |

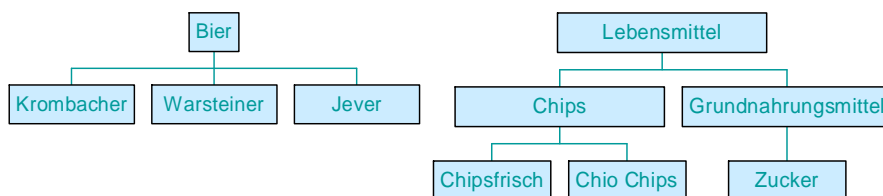
18

Taxonomien: Generalisierten Assoziationsregeln

- Häufig sind Assoziationsregeln recht redundant.
- Assoziationen zwischen Produktgruppen können schlecht ausgedrückt werden.
- Wenn z.B. jeder Kunde, der Chips kauft, auch Bier kauft, dann finden sich Regeln
 - ◆ Chio Chips \Rightarrow Krombacher
 - ◆ Chio Chips \Rightarrow Warsteiner
 - ◆ Chipsfrisch Chips \Rightarrow Jever
- Idee: Abstraktion durch Taxonomien

19

Taxonomien



- „Chips \Rightarrow Bier“
- „Jever \Rightarrow Chipsfrisch“

20

Generalisierte Assoziationsregeln: Problemstellung

- Gegeben
 - ◆ Menge I von Items,
 - ◆ Menge T von Transaktionen; $t \in T \Rightarrow t \subseteq I$.
- Azyklischer Graph TAX über Knoten I.
 - ◆ $(p,c) \in \text{TAX}$: p ist allgemeiner als c.
 - ◆ TAX*: Transitive Erweiterung von TAX.
- Finde alle Regeln $r = (X \Rightarrow Y)$ mit $X \subseteq I, Y \subseteq I$, so dass
 - ◆ Support $s(r) \geq s_{\min}$
 - ◆ Confidence $c(r) \geq c_{\min}$

- Wobei

$$s(X) = \frac{|\{t \in T \mid \forall x \in X : x \in t \vee (\exists y : (x, y) \in \text{TAX}^* \wedge y \in t)\}|}{|T|}$$

21

„Cumulate“-Algorithmus

- Häufige-Itemsets (s_{\min}, TAX)
- 1. Bestimme TAX*
- 2. C_1 = häufige Itemsets Größe 1; k=2
- 3. Solange $L_k \neq \{ \}$
 1. C_{k+1} = Generiere Kandidaten
 2. Wenn k=2, dann eliminiere Kandidaten, die aus einem Item und seinem Vorgänger bestehen
 3. Lösche alle $c \in \text{TAX}^*$ für die kein $(c,p), p \in c, c \in C_{k+1}$ existiert.
 4. Für alle $t \in T$, für alle $p \in t$, füge alle c mit $(c,p) \in \text{TAX}^*$ zu t hinzu, entferne Duplikate.
 5. Erhöhe Zähler für alle Kandidaten c, die in t und in C_{k+1} vorkommen.
 6. L_{k+1} = Prune (C_{k+1}); Erhöhe k.
- 4. Rückgabe ist Vereinigung aller L_k .

22