

Data Mining: Association Rules

- Sei $\mathcal{I} = \{I_1, I_2, \dots\}$ eine Menge von *Items*. $X \subseteq \mathcal{I}$ nennen wir *Itemset*.
- Sei $T \subseteq \mathcal{I}$ ein *Itemset*, der eine (*Kunden*)-*Transaktion* repräsentiert, z.B. einen Einkauf der gerade die Items in T betrifft.
- Die Menge der zu betrachtenden Transaktionen sei in einer Tabelle Purchases(Trans_Id, Item) gespeichert.
- Seien $X, Y \subseteq \mathcal{I}$, $X \cap Y = \emptyset$. Der Ausdruck $X \Rightarrow Y$ heißt *Association Rule* und drückt aus, dass wenn X Teil einer Transaktion, dass dann auch Y Teil dieser Transaktion.

Beispiel Verkaufstabelle

<u>Transaction_Id</u>	<u>Product</u>
001	diapers
001	beer
001	popcorn
001	bread
002	diapers
002	cheese
002	soda
002	beer
002	juice
003	diapers
003	cold cuts
003	cookies
003	napkins
004	cereals
004	beer
004	cold cuts

Problem

- Bestimme die Itemsets X mit einem *Support* $supp(X)$ größer einer vorgegebenen Schranke s ; der Support ist definiert als der Prozentsatz der Transaktionen, deren Itemset X enthält. Solche Itemsets nennen wir *frequent*.
- Bestimme die Association Rules $X \Rightarrow Y$, für die XY frequent ist und deren *Confidence* größer einer vorgegebenen Schranke c ist; die Confidence ist definiert als der Prozentsatz der Transaktionen, die zusätzlich zu X auch Y enthalten.

Beispiel

- Association Rule: $Purchase_diapers \Rightarrow Purchase_beer$
- Support: 50%
- Confidence: 66,66%

a Priori Property

Every subset of a frequent itemset is also a frequent itemset.

Algorithm for Finding Frequent Itemsets

```
foreach item, check if it is a frequent itemset;
k = 1;
repeat
  foreach new frequent itemset I with k items
    extend it to all itemsets I' with k+1 items;
  Scan all transactions once and check if the generated
  itemsets I' are frequent;
  k = k + 1;
until no new frequent itemsets are identified;
```

Bemerkungen

- Es müssen nur solche Itemsets I' mit $k + 1$ Items für den Scan der Transaktionen berücksichtigt werden, deren Teilmengen mit k Items frequent sind. Sei $I' = I \cup \{T\}$; teste alle solche Teilmengen, die T enthalten - diese Information ist bereits vorhanden.

Finding Association Rules

- Betrachte $X \Rightarrow Y$, wobei $\text{supp}(XY) \geq s$.
- Teste $\frac{\text{supp}(XY)}{\text{supp}(X)} \geq c$.
- $\text{supp}(XY)$ und $\text{supp}(X)$ wurden bereits bestimmt.

Data Mining: Klassifikation mittels Entscheidungsbäumen

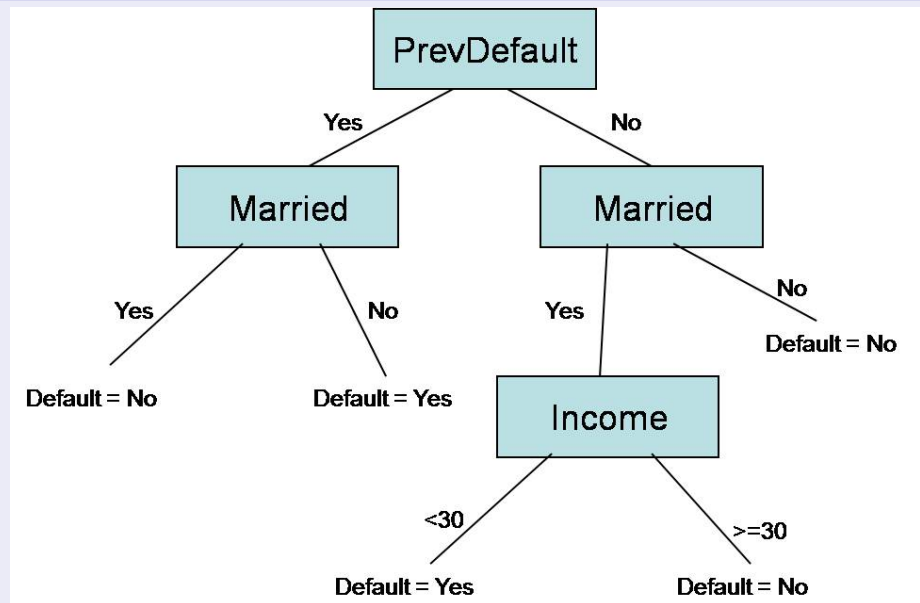
Klassifikation

- Die interessierenden Kategorien sind bekannt und in Beispielen dokumentiert (*supervised learning*).
- Finde Muster unter ausgewählten Attributwerten (*Predictor-Attribute*) der einzelnen Sätze einer gegebenen Datenbank (*Training Records*).
- Ordne in Abhängigkeit von den konkreten Mustern die Tupel in Kategorien ein (*Class-Label*), u.U. unter Inkaufnahme eines *Training-Set Errors*.
- Verwende diesen Zusammenhang für den Aufbau eines Entscheidungsbaumes der die Vorhersage der Kategorie von Tupeln ermöglicht, deren Attributwerte bekannt, jedoch deren Kategorie als unbekannt angenommen werden (*Test Records*); mögliche Fehlklassifikationen definieren den *Test-Set Error*.
- Ist der Test-Set Error erheblich größer als der Training-Set Error, liegt u.U. ein *Overfitting* vor. In diesem Fall wird die Höhe des Baumes verringert durch Pruning von Teilbäumen.

Beispiel Kundentabelle bzgl. Hypotheken-Vergabe

Id	Married	Prevdefault	Income	Default
C1	yes	no	50	no
C2	yes	no	100	no
C3	no	yes	135	yes
C4	yes	no	125	no
C5	yes	no	50	no
C6	no	no	30	no
C7	yes	yes	10	no
C8	yes	no	10	yes
C9	yes	no	75	no
C10	yes	yes	45	yes
C11	yes	no	60	no
C12	no	yes	125	yes
C13	yes	yes	20	no
C14	no	no	15	no
C15	no	no	60	no
C16	yes	no	15	yes
C17	yes	no	35	no
C18	no	yes	160	yes
C19	yes	no	40	no
C20	yes	no	30	no

Entscheidungsbaum



einige Klassifikations-Regeln

- (1) $((\text{PrevDefault}=\text{yes}) \text{AND} (\text{Married}=\text{yes})) \Rightarrow (\text{Default}=\text{no})$
- (2) $((\text{PrevDefault}=\text{no}) \text{AND} (\text{Married}=\text{yes})) \text{AND} (\text{Income} < 30) \Rightarrow (\text{Default}=\text{yes})$
- (3) $((\text{PrevDefault}=\text{no}) \text{AND} (\text{Married}=\text{yes})) \text{AND} (\text{Income} \geq 30) \Rightarrow (\text{Default}=\text{no})$

Training-Set Error: z.B. Regel (1) auf C10 angewendet liefert fälschlicherweise no.

Entscheidungsbaum T

- Jedem Blatt ist ein Class-Label zugeordnet.
- Jedem inneren Knoten ist ein Predictor-Attribut zugeordnet (*Splitting-Attribut*).
- Jeder Kante e von einem inneren Knoten n ist ein Prädikat über dem n zugeordneten Predictor-Attribut zugeordnet.
- Die den ausgehenden Kanten eines inneren Knotens zugeordneten Prädikate (*Splitting-Prädikate*) sind erschöpfend, d.h. ihre Disjunktion ergibt `true` und nicht-überlappend, d.h. die Konjunktion von je zwei Prädikaten ergibt `false`.
- Das Splitting-Attribut und die Splitting-Prädikate eines Knotens n ergibt das Splitting-Kriterium $crit(n)$.

Prinzip des Aufbaus eines Entscheidungsbaumes

- Beginnend mit dem der Wurzel des Baumes zugeordneten Attribut A (im Beispiel *PrevDefault*) wird rekursiv zu dem betrachteten ein Attribut A' für den nächst höheren Level des Baumes ausgewählt.
- Sei K der Knoten des betrachteten Attributes und seien K'_1, \dots, K'_r in Abhängigkeit der möglichen Werte a_1, \dots, a_r von A' in der Trainingsmenge die A' zugeordneten Knoten. (Ist z.B. *Income* das betrachtete Attribut, so ergeben sich Anzahl unterschiedlicher *Income*-Werte viele Knoten K').
- Den Kanten $K \rightarrow K'_i$ wird der Attributwert a_i zugeordnet. Die Anzahl Knoten kann verringert werden, indem Ausdrücke über den möglichen Attributwerten (z.B. *Income* < 30) verwendet werden.
- Setze dieses Verfahren so lange fort, bis alle Blätter im Baum eine Klassifizierung der zu betrachtenden Sätze erlauben.

Auswahl eines Attributes

Das ausgewählte Attribut soll möglichst Teilbäume definieren, deren Blätter jeweils möglichst viele Sätze derselben Klasse definieren.

Entropie als Grundlage

- Wenn ein Ereignis, das mit Wahrscheinlichkeit p eintreten kann, tatsächlich eintritt, dann wird dadurch ein konkretes Ereignis aus einer hypothetischen Menge von $\frac{1}{p}$ von gleich wahrscheinlichen Ereignissen ausgewählt.
- Um diese Anzahl von Ereignissen unterscheiden zu können benötigt man $\log_2\left(\frac{1}{p}\right) = -\log_2(p)$ Binärbits.
- Je kleiner die Auftrittswahrscheinlichkeit p eines Zeichens ist, desto höher ist seine Information.
- *Entropie* $H(X)$, X Zufallsvariable über einem abzählbaren Alphabet $Z = \{z_1, z_2, \dots\}$ mit Wahrscheinlichkeit $P(X \in Z) = 1$ und $p_i = P(X = z_i)$ ist definiert zu

$$H(X) = - \sum_{i=1}^{|Z|} p_i \log_2(p_i)$$

- Die Entropie ist Erwartungswert der Zeichenlänge in Bit (vergl. Huffman-Kodierung).

Beispiele

- Das Alphabet Z ist gegeben durch die Klassen der betrachteten Sätze. Die Wahrscheinlichkeiten ergeben sich aus den relativen Häufigkeiten.
- Tritt in den Sätzen $Default = No$ und $default = Yes$ mit gleicher Häufigkeit auf:

$$-(1/2 \log_2 1/2 + 1/2 \log_2 1/2) = 1$$

Maximal mögliche Entropie, minimaler Informationsgehalt da maximale Ungewissheit.

- In den Sätzen tritt ausschließlich $Default = No$ oder $default = Yes$ auf:

$$-\log_2 1 = 0$$

Minimal mögliche Entropie, maximaler Informationsgehalt da maximale Gewissheit.

- In der Tabelle finden sich 6 Yes-Sätze und 14 No-Sätze:

$$(6/20 \log_2 6/20 + 14/20 \log_2 14/20) = 0.881$$

Betrachte das Predictor-Attribut *PrevDefault*. Die Menge der Sätze zu *Yes* hat 6 Elemente, davon 4 mit *Default = Yes* und 2 mit *Default = No*:

$$(4/6 \log_2 4/6 + 2/6 \log_2 2/6) = 0.918$$

Die Menge der Sätze zu *No* hat 14 Elemente, davon 2 mit *Default = Yes* und 12 mit *Default = No*:

$$(2/14 \log_2 2/14 + 12/14 \log_2 12/14) = 0.592$$

Die *gewichtete Entropie* ist damit

$$(6/20 * 0.918) + (14/20 * 0.592) = 0.69$$

Splitting-Strategie

Wähle als Predictor-Attribut ein Attribut *A* so, dass der *Information Gain* maximal ist:

$$\text{InformationGain}(A, T) = \text{entropy}(T) - \text{sum}(\text{entropy}(T_i) * \text{weight}(T_i))$$

Die einzelnen T_i sind Teilmengen von T ; $\text{weight}(T_i) = \frac{|T_i|}{|T|}$.

$$\text{InformationGain}(\text{PrevDefault}, T) = 0.881 - 0.69$$

Entwickle Verfahren, die auch für externen Datenbanken anwendbar sind, die nicht im Hauptspeicher gehalten werden können.

Auswahl der Sätze mittels SQL.

Attribute-Value, Class-Label; AVC-Set.

- Sei C Class-Label, n ein Knoten und X ein Predictor-Attribut von n .
- Der AVC-Set $AVC_n(X)$ ist das Resultat des SQL-Ausdrucks:

```
SELECT D.X, D.C, COUNT(*)  
FROM D  
WHERE F(n)  
GROUP BY D.X, D.C
```

- Die Größe eines AVC-Set ist unabhängig von der Größe der Datenbank D ; sie hängt von der Anzahl unterschiedlicher Werte des Predictor-Attributs und der Anzahl Class-Labels $F(n)$ von der Wurzel des in der Konstruktion befindlichen Baumes zu n ab.
Bei einem kontinuierlichen Wertebereich eines Attributs - vergl. *Income* - kann ein AVC-Set in Abhängigkeit von den in der Datenbank vorhandenen Werten entsprechend viele Sätze pro Class-Label enthalten.