

**Übung zur Vorlesung
„Einsatz und Realisierung von Datenbanksystemen“
im Sommersemester 2007**

Richard Kuntschke (richard.kuntschke@in.tum.de)

Lösungen zu Blatt 8

Aufgabe 1

Klassifikationsbäume können durch ein *top-down*-Verfahren konstruiert werden. Man startet mit der gesamten Datenbasis als initiale Partition für den Wurzelknoten. In jedem Schritt wird dann untersucht, ob die jeweils aktuelle Partition sinnvoll weiter unterteilt werden kann. Dazu bestimmt man ein Vorhersage-Attribut und einen Attributwert, die die Aufteilung in zwei kleinere Teil-Partitionen bestimmen. Benötigt wird dazu ein Entscheidungsalgorithmus, der mögliche Aufspaltungen vergleicht und evaluiert, ob und falls ja, welche Partitionierung sinnvoll ist. Dabei bestimmt sich „sinnvoll“ in Hinblick auf die Herleitung von aussagekräftigen Ableitungsregeln.

Eine Partition, die dem Knoten n im Klassifikationsbaum zugeordnet ist, wird somit in zwei Teil-Partitionen unterteilt. n werden zwei Kindknoten zugewiesen¹. Eine der Partitionen wird dem linken und die andere dem rechten Kind zugewiesen. Die Partitionierung führt man dann rekursiv auf den Kindknoten fort.

Für jeden Partitionierungsschritt sind während der Ausführung also zwei Dinge zu wählen:

1. das Vorhersage-Attribut gemäß dem die Aufspaltung erfolgen soll und
2. der Attributwert, der die Partitionen voneinander trennt.

Wir wollen im Folgenden weniger auf Algorithmen zur Entscheidungsfindung eingehen als uns vielmehr auf die dabei zu analysierenden Daten konzentrieren. Eine Herausforderung für die algorithmische Durchführung stellt die Größe der zu betrachtenden Partitionen dar. So muss beispielsweise für den Wurzelknoten die gesamte Datenbasis betrachtet werden, die bei für Data Mining-Anwendungen typischerweise großen Ausprägungen nicht vollständig in den Hauptspeicher geladen werden kann. Da aber jedes Vorhersage-Attribut getrennt betrachtet wird, kann man das Laden der gesamten Datenbasis (bzw. der Partition) umgehen und sich stattdessen auf aggregierte Ergebnisse beschränken.

In unserem Beispiel zur Risikoanalyse gibt es drei Kandidaten für die zu wählenden Vorhersage-Attribute, nämlich *wiealt*, *Geschlecht* und *Autotyp*. Für jedes dieser Attribute erstellen wir Klassifikations-Aggregate zur Risikoabschätzung:

```
select wiealt, Schäden, count(*)  
from Schadenshöhe  
group by wiealt, Schäden
```

```
select Autotyp, Schäden, count(*)  
from Schadenshöhe  
group by Autotyp, Schäden
```

¹Es gibt auch Verfahren für nicht-binäre Klassifikationsbäume.

```
select Geschlecht, Schäden, count(*)
from Schadenshöhe
group by Geschlecht, Schäden
```

Ergebnisse derartiger Aggregatberechnungen werden in der Literatur auch als AVC-Mengen (*Attribute Value Class*) bezeichnet. Die AVC-Mengen für die Partition des Wurzelknotens, d.h. die gesamte Datenbasis, sind in unserem Beispiel:

wiealt	Schäden	
	gering	hoch
18	1	0
19	0	1
22	1	0
24	0	1
38	1	0
40	1	1
45	0	1

Autotyp	Schäden	
	gering	hoch
Coupé	2	2
Van	3	1

Geschlecht	Schäden	
	gering	hoch
w	4	0
m	1	3

Die Größe einer AVC-Menge hängt von der Anzahl der unterschiedlichen Werte für ein Vorhersage-Attribut der Partition ab. In unserem Beispiel haben die AVC-Menge für *wiealt* 7 und die AVC-Mengen für *Autotyp* und *Geschlecht* jeweils 2 Einträge. Die in Abbildung 1 gezeigte Ausprägung hat 8 Beispiel-Tupel. Für große Datenbanken ist die Mächtigkeit der AVC-Mengen unabhängig von der Anzahl der Tupel in der Datenbasis – ausgenommen AVC-Mengen für Vorhersage-Attribute, deren Domänen sehr groß sind, wie beispielsweise Fließkommazahlen.

Wenn wir annehmen, dass die AVC-Mengen für die Datenbasis in den Hauptspeicher passen, können wir den Algorithmus zur Berechnung von Klassifikationsbäumen in Pseudo-Code wie folgt angeben:

1. Erzeuge die AVC-Mengen für die aktuelle Partition.
2. Versuche eine gute Partitionierung zu finden. Falls ein Vorhersage-Attribut und ein Wert für eine sinnvolle Partitionierung bestimmt wird:
 - Erzeuge zwei Kindknoten n_1 und n_2 im Klassifikationsbaum.
 - Teile die aktuelle Partition auf in P_1 und P_2 .
 - Weise P_1 n_1 und P_2 n_2 zu.
 - Führe die Partitionierung rekursiv für die beiden Kindknoten n_1 und n_2 fort.

Aufgabe 2

Siehe Buch “Datenbanksysteme”, Kapitel 17.2.9.

Aufgabe 3

Die *confidence* einer Assoziationsregel $L \Rightarrow R$ gibt die Übereinstimmung der Regel bezüglich der Datenbasis an. Sie bestimmt sich aus dem Umfang der Datenmenge, die die Voraussetzung L und die Schlussfolgerung R erfüllt, relativ zur Größe der Menge, die die Voraussetzung L erfüllt.

Behauptung: Wenn $L \subseteq L^+$, $R^- \subseteq R$ und $L \cup R = L^+ \cup R^- = F$ gilt

$$confidence(L^+ \Rightarrow R^-) \geq confidence(L \Rightarrow R)$$

Schadenshöhe			
wiealt	Geschlecht	Autotyp	Schäden
45	w	Van	gering
18	w	Coupé	gering
22	w	Van	gering
38	w	Coupé	gering
19	m	Coupé	hoch
24	m	Van	hoch
40	m	Coupé	hoch
40	m	Van	gering
⋮	⋮	⋮	⋮

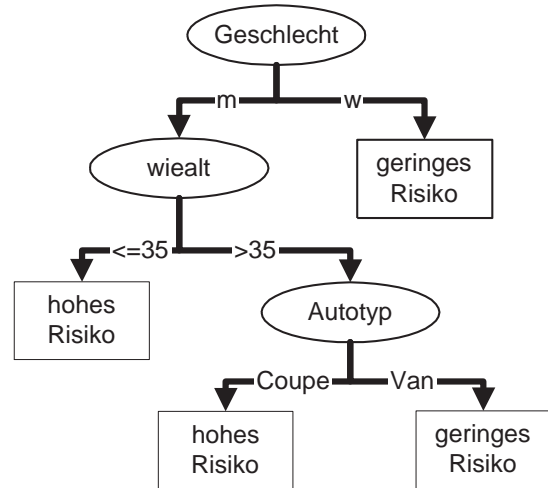


Abbildung 1: Klassifikationsschema für Haftpflicht-Risikoabschätzung

Beweis: Sei $\Delta \subset F$ für die gemäß der Angabe gilt $\Delta = L^+ - L = R - R^-$.

Daraus folgt: $L^+ = L \cup \Delta$ und $R^- = R - \Delta$

$$\begin{aligned}
 \text{confidence}(L^+ \Rightarrow R^-) &= \frac{\text{support}(L^+ \cup R^-)}{\text{support}(L^+)} \\
 &= \frac{\text{support}(\{L \cup \Delta\} \cup \{R - \Delta\})}{\text{support}(L \cup \Delta)} \\
 &= \frac{\text{support}(L \cup R)}{\text{support}(L \cup \Delta)} \\
 &\geq \frac{\text{support}(L \cup R)}{\text{support}(L)} \\
 &= \text{confidence}(L \Rightarrow R)
 \end{aligned}$$

Die Ungleichung gilt, da $\text{support}(L \cup \Delta) \leq \text{support}(L)$, da der Support einer Menge L gleich bleibt oder reduziert wird, wenn L vergrößert wird: Es gibt dann weniger oder höchstens gleich viele Einträge in der Datenbasis, die die durch Δ beschriebenen zusätzlichen Bedingungen erfüllen.

Aufgabe 4

Abbildung 3 zeigt den à priori-Algorithmus in Pseudo-Code. Gemäß dem Beispiel ist $\text{minsupp} = 3/5$. Im dritten Schritt ergibt sich nur noch für die Menge {Drucker, Papier, Toner} ein support von $3/5$, alle andere Mengen müssen also nicht weiter betrachtet, d.h. erweitert werden. Mögliche weitere Artikel sind: PC und Scanner. Scanner scheiden jedoch von vornherein aus, da deren support nur bei $2/5$, also unterhalb minsupp , liegt. Da der support von {Papier, PC} = $2/5$ und {Papier, PC} \subset {Drucker, Papier, Toner, PC} ist, fällt auch diese Erweiterung unterhalb die geforderte Schranke minsupp . Somit terminiert der Algorithmus und liefert *frequent itemsets* der maximalen Größe 3.

VerkaufsTransaktionen		Zwischenergebnisse	
TransID	Produkt	FI-Kandidat	Anzahl
111	Drucker	{Drucker}	4
111	Papier	{Papier}	3
111	PC	{PC}	4
111	Toner	{Scanner}	2
222	PC	{Toner}	3
222	Scanner	{Drucker, Papier}	3
333	Drucker	{Drucker, PC}	3
333	Papier	{Drucker, Scanner}	3
333	Toner	{Drucker, Toner}	3
444	Drucker	{Papier, PC}	2
444	PC	{Papier, Toner}	3
555	Drucker	{PC, Scanner}	2
555	Papier	{PC, Toner}	2
555	PC	{Scanner, Toner}	2
555	Scanner	{Drucker, Papier, PC}	3
555	Toner	{Drucker, Papier, Toner}	3
		{Drucker, PC, Toner}	3
		{Papier, PC, Toner}	3

Abbildung 2: Datenbank mit Verkaufstransaktionen (links) und Zwischenergebnisse des à priori-Algorithmus (rechts)

- **für alle** Produkte
 - überprüfe, ob es ein *frequent itemset* ist, also in mindestens *minsupp* Einkaufswägen enthalten ist
 - $k := 1$
 - **iteriere so lange**
 - **für jeden** *frequent itemset* I_k mit k Produkten
 - * generiere alle *itemsets* I_{k+1} mit $k + 1$ Produkten und $I_k \subset I_{k+1}$
 - lies alle Einkäufe einmal (sequentieller Scan auf der Datenbank) und überprüfe, welche der $(k + 1)$ -elementigen *itemset*-Kandidaten mindestens *minsupp* mal vorkommen
 - $k := k + 1$
- bis keine neuen** *frequent itemsets* gefunden werden

Abbildung 3: Der à priori-Algorithmus