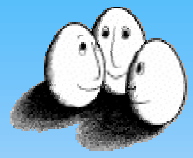
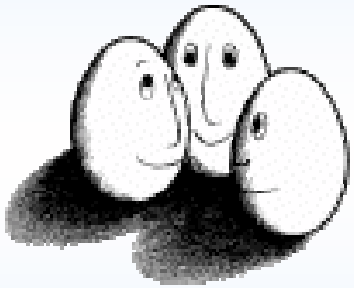


Apriori-Algorithmus zur Entdeckung von Assoziationsregeln



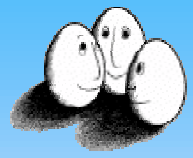
PG 402 Wissensmanagment

Lehrstuhl für Künstliche Intelligenz

22. Oktober 2001

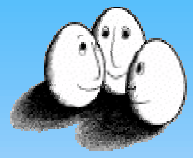
Zoulfa El Jerroudi

Gliederung



- Motivation
- Formale Problemdarstellung
- Apriori-Algorithmus
- Beispiel
- Varianten des Apriori-Algorithmus
- Weka
- Zusammenfassung

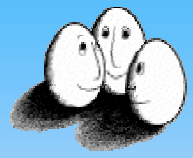
1. Motivation



- Interesse an noch unbekannten Zusammenhängen in einer großen DB
- Beispiel: Supermarktkette möchte für die Marketingstrategie wissen, welche Artikel zus. gekauft werden. Z.B.: In 67% der Fälle in denen Cola u. Saft zus. gekauft werden, wird auch Bier gekauft
- Eine Assoziationsregel ist ein Ausdruck X => Y

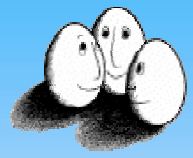
  (wobei X u. Y Mengen von Objekten sind)
- Bedeutung: Transaktionen der DB, die X enthalten, tendieren dazu auch Y zu enthalten

2. Formale Problemdarstellung



- Es sei **$I=\{i_1, i_2, ...,i_n\}$ eine Menge von Objekten** (Items)
- **D** sei eine **Menge von Transaktionen**, wobei jede Transaktion eine Menge von Objekten ist, also T ? I. Mit jeder Transaktion ist ein eindeutiger Bezeichner, genannt TID assoziiert.
- Eine **Assoziationsregel** ist eine Implikation der Form **X=>Y**, wobei X und Y Untermengen von I sind. (Und X und Y keine gemeinsamen Elemente haben)
- Eine Regel X=>Y hat den **Konfidenzwert c**, falls c% der Transaktionen aus D, die X enthalten auch Y enthalten.
- Eine Regel X=>Y hat den **Support s**, wenn s% der Transaktionen aus D, X vereinigt Y enthalten.

2.1 Das Problem



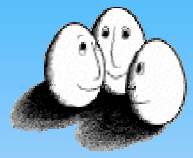
- Bei einer gegebenen Datenmenge D von Transaktionen besteht das Problem der Entdeckung der Assoziations-beziehungen darin, alle Assoziationsregeln X=>Y mit

  Konfidenz(X=>Y) >= min. Konfidenz und

  Support(X=>Y) >= min. Support abzuleiten.

- **Beispiel:** Der Benutzer gibt als minimalen Konfidenzwert 70 ein. Dann sollen alle Regeln X=>Y abgeleitet werden, falls 70% der Transaktionen die X enthalten auch Y enthalten.

2.2 Zwei Teilprobleme



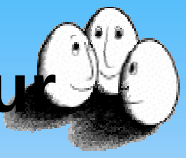
1. Finde alle Kombinationen von Objekten mit

   Support >= min.Support. Diese Kombinationen werden *große* Objektmengen genannt, alle anderen Kombinationen *klein*.

2. Benutze die großen Objektmengen um die gewünschten Regeln zu erzeugen. Die Idee: jede große Objektmenge wird in jeweils zwei disjunkte Teilmengen X und Y aufgeteilt, um Assoziationsregeln der Form X=>Y zu generieren. Wenn die Beziehung Konfidenz(X=>Y) > c_{min} gilt,

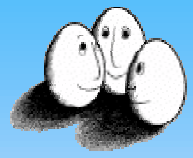
   dann wird die Regel X=>Y ausgegeben.

2.3 Vorgehensweise des Algorithmus zur Entdeckung *großer* Objektmengen



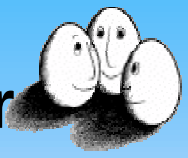
- 1. Iteration: häufig vorkommenden Objekte werden bestimmt. D.h. man zählt für jedes Objekt, wie häufig es in den Trans-aktionen vorkommt,
- Danach werden die aus der vorherigen Iteration entdeckten *großen* Objektmengen als Grundmenge benutzt, um neue potentielle *große* Objektmengen zu erzeugen, genannt **Kandidaten**.
- Es wird der aktuelle Support für diese Kandidaten ermittelt. Am Ende eines Durchlaufs wird geprüft, welche Kandidaten wirklich groß sind. Diese werden zur Grundmenge der nächsten Iteration.
- Prozess wird solange wiederholt, bis keine neuen großen Objektmengen mehr gefunden werden.

3. Apriori-Algorithmus



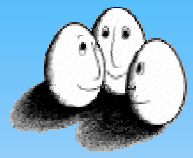
- G_1 = {große Objektmenge}
- **FOR** (k=2; G_{k-1} != { }; k++) **DO BEGIN**
- C_k = apriori-gen(G_{k-1}*);* *neue Kandidaten werden erzeugt*
- **FORALL** transaction t ? D **DO**
- **BEGIN**
- C_t = subset(C_k,t) **;** *Kandidaten enthalten in t*
- FORALL Kandidaten c ? C_t DO
- c.support= c.support +1;
- **END**
- G_k = {c ? C_k I c.support >= minsupport}
- **END**
- **Ausgabe**: ☠$_k$ G_k
- Regelgenerierung aus G_k

3.1 Apriori-gen-Methode zur Erzeugung der Kandidatenmenge



- **Eingabe** der Methode: G_{k-1}, die Menge der großen Objektmengen mit k-1 Elementen
- **Rückgabe**: Teilmenge der Menge aller großen Objektmengen mit k Elementen
- Im 1.Schritt zwei Mengen mit k-1 Elementen werden vereinigt und zu den Kandidaten hinzugefügt, wenn k-2 Elemente der Mengen übereinstimmen.
- Im nächsten Schritt entferne aus den Kandidaten mit k Elementen, die Mengen für die die Untermengen mit k-1 Elementen nicht zu den großen Objektmengen gehören.
- Wir können deshalb so vorgehen, weil jede Untermenge einer großen Objektmenge, selbst wieder groß ist.

4. Beispiel

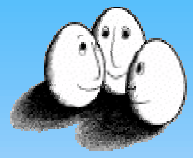


Transaktion	Objekt (Artikel)
t_0	Saft, Cola, Bier
t_1	Saft, Cola, Wein
t_2	Saft, Wasser
t_3	Cola, Bier
t_4	Saft, Cola, Bier,Wein
t_5	Wasser
t_6	Schokolade, Cola, Chips
t_7	Schokolade, Schinken, Brot
t_8	Brot, Bier

Benutzereingabe:

minimaler Support =2
minimaler Konfidenzwert= 75%

4.1 Die Kandidatenmenge



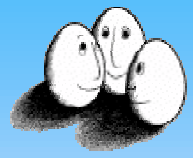
- Nach dem ersten Durchlauf der Transaktionen besteht die Kandidatenmenge C_1 aus:

Objektmenge	**Support**
{Cola}	5
{Saft}	4
{Bier}	4
{Wein}	2
{Wasser}	2
{Schokolade}	2
{Brot]	2
~~{Chips}~~	~~1~~
~~{Schinken}~~	~~1~~

Die große Objektmenge enthält diejenigen Kandidaten, die einen Supportwert von mindestens 2 aufweisen

G_1={ {Cola},{Saft}, {Bier}, {Wein}, {Wasser}, {Schokolade}, {Brot} }

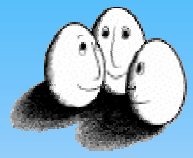
4.2 Kandidatenmenge C_2



- Aus G_1 werden alle Kombinationen gebildet um Kandidatenmengen mit zwei Elemente zu erhalten.

Objektmenge	**Support**	**Objektmenge**	**Support**
[Cola, Saft}	3	{Bier, Wein}	1
{Cola,Bier}	3	{Bier, Wasser}	0
{Cola, Wein}	2	{Bier, Schokolade}	0
{Cola, Wasser}	0	{Bier, Brot}	1
{Cola,Schokolade}	1	{ Wein, Wasser}	0
{Cola, Brot}	0	{Wein, Schokolade}	0
{Cola, Bier}	2	{Wein, Brot}	0
{Saft, Wein}	2	{Wasser, Schokolade}	0
{Saft, Wasser}	1	{Wasser, Brot}	0
{Saft, Schokolade}	0	{Schokolade, Brot}	1
{Saft, Brot}	0		

4.3 Kandidatenmenge C_3

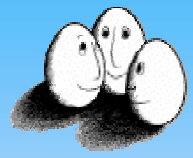


- **G_2** = {c ? C_2 I c.support > min.support}

  G_2 = {{Cola,Saft}, {Cola,Bier}, {Cola,Wein}, {Saft,Bier}, {Saft,Wein}}
- 2.Iteration abgeschlossen
- 3.Iteration:
  - C_3= { a ☠ b I a,b ? L_2 und Ia ⚐ b I = 1}
  - {Cola, Saft} und {Cola Bier} werden zu {Cola, Saft, Bier}
  - {Cola, Bier} und {Cola, Wein} werden zu {Cola, Bier, Wein}

    kommt jedoch nicht in C_3, da Teilmenge {Bier, Wein} keine große Objektmenge ist
  - **C_3 = {{Cola, Saft, Bier}, {Cola, Saft, Wein}}**

4.4 Letzte Iteration



- Kandidatenmenge C_3

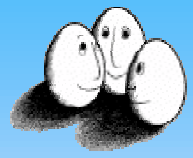
Objektmenge	Support
{Cola,Saft,Bier}	2
{Cola,Saft,Wein}	2

G_3={{Cola,Saft,Bier} {Cola,Saft,Wein}}

- {Cola, Saft, Bier, Wein} ist nicht in C_4, da Teilmenge {Saft, Bier, Wein} nicht groß ist.

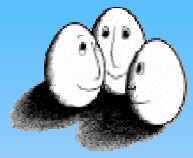
- C_4 und G_4 sind leer und somit stoppt der Algorithmus

5. AprioriTID-Algorithmus



- Variante des Apriori-Algorithmus
- eigentliche Datenmenge wird nach der 1.Iteration nicht mehr zum Zählen der Supportwerte der Kandidaten benutzt.
- Dazu, eine spezielle Menge, in der jede Transaktion durch eine Menge darin vorkommenden Kandidaten ersetzt wird.
- In späteren Iterationen kann die Anzahl der Einträge in dieser speziellen Menge geringer sein als die Anzahl der Trans. in D
- Außerdem kürzer als die korrespondierende Trans., weil in späteren Iterationen wahrscheinlich nur sehr wenige Kandidaten in der Trans. enthalten sind.
- Jedoch wird in früheren Iterationen sehr viel Speicher benötigt, da jeder Eintrag in der speziellen Menge durch die Aufnahme aller in der Trans. vorkommenden Kandidaten sehr viel größer als die Trans. sein kann.

5.1 AprioriHybrid-Algorithmus



- Nach diesen Beobachtungen hat man einen Algorithmus konstruiert, der in der Anfangsphase den Apriori-Algoritmus, benutzt, und später, wenn zu erwarten ist, dass die spezielle Menge in den Speicher passt, den AprioriTID-Algorithmus.

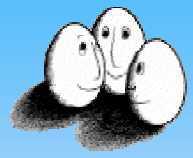
- Der AprioriHybrid-Algorithmus liefert nur dann schlechtere Ergebnisse als der Apriori-Algorithmus, wenn der Wechsel in den letzten Iterationen passiert. Dann nimmt er die Kosten des Wechsels auf sich (die Generierung der speziellen Menge) ohne die Vorteile auszunutzen.

6. Weka (Waikato Enviroment for Knowledge Analysis)

- Sammlung von Java-Paketen
- Implementiert Algorithmen aus dem Bereich Data Mining und maschinelles Lernen
- Im Netz frei erhältlich
- Enthält Tools für:
  - das Preprocessing der Daten
  - das Einbinden in ein Lernverfahren
  - die Analyse der Ergebnisse

- Enthält Implementierung des Apriori-Algorithmus

7. Zusammenfassung



- Apriori-Algor. Zur Entdeckung von Asoziationsregeln
- Vorgehensweise in 2 Schritten
  - Bestimmung der großen Objektmengen
  - Benutzung der großen Objektmengen zur Regelgenerierung. Jede Menge wird in zwei disjunkte Teilmengen X und Y aufgeteilt. Assoziationsregel: X=>Y
- Laufzeit: O(ICI IDI)
  - ICI : Summe der Größen der Kandidaten
  - IDI : Größe der Datenmenge
- AprioriTID braucht in den frühen Iterationen viel Speicherplatz, spart aber in den letzten
- Deshalb Einführung des AprioriHybrid
  - Benutzt in der Anfangsphase den Apriori und später den AprioriTID-Algorithmus