数据挖掘
第六讲·频繁模式挖掘

购物篮分析问题

- 引例：购物篮分析
  - 零售商通过发现顾客放入“购物篮”中的不同物品之间的**关联**，分析顾客的购买模式。
  - 帮助零售商了解哪些商品频繁地被顾客同时购买，从而帮助他们开发更好的营销策略。

购物篮分析问题

- 引例：购物篮分析
  - 案例一：啤酒+尿布

WAL★MART 沃尔玛

- 案例二：购物网站产品推荐

亚马逊 amazon.cn

频繁模式（Frequent pattern）

{牛奶，面包}：项集

PC→ 数码相机→内存卡：序列模式

子图、子树等子结构：结构模式

频繁模式

频繁出现在数据集中的模式（项集、子序列或子结构）

经常一起购买的商品摆放近一点，促进两种商品的购买。

硬件和软件摆放在商店的两端，促进杀毒软件、家庭安全系统的购买。

计算机和打印机经常一起购买，打印机降价同时促进二者购买。

频繁模式挖掘：数据库、数据挖掘领域的重要问题

	Title	Authors	Published in	#Cit.
1	Fast Algorithms for Mining Association Rules	R. Agrawal, R. Srikant	VLDB '94	2261
2	Querying Heterogeneous Information Sources Using Source Descriptions	A.Y. Levy, A. Rajaraman, J.J. Ordille	VLDB '96	692
3	BIRCH: An Efficient Data Clustering Method for Very Large Databases	T. Zhang, R. Ramakrishnan, M. Livny	SIGMOD '96	617
4	Mining Frequent Patterns without Candidate Generation	J. Han, J. Pei, Y. Yin	SIGMOD '00	573
5	Implementing Data Cubes Efficiently	V. Harinarayan, A. Rajaraman, J.D. Ullman	SIGMOD '96	559

The top 5 most referenced DB conference publications (1994-2003), adapted from "Citation analysis of database publication", SIGMOD Record, 34(4), Dec. 2005

提纲

关联规则基本概念

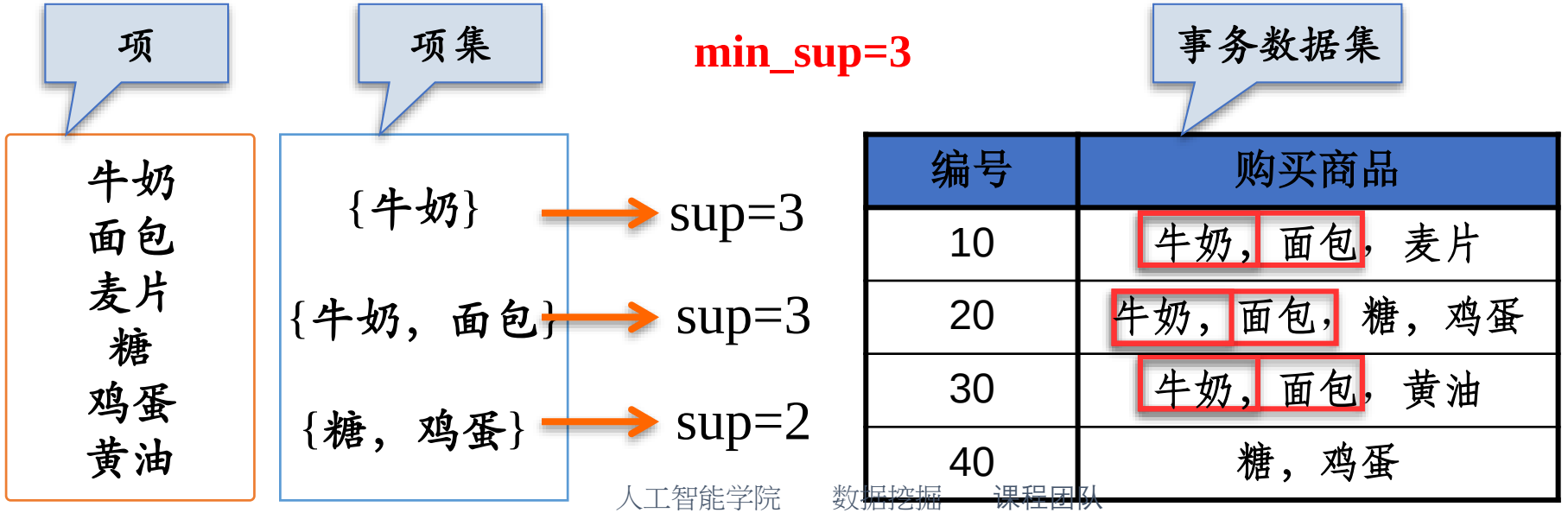
- 关联规则形式
  - A → B [support, confidence]
  - 支持度(support)和置信度(confidence) 是规则兴趣度的两种度量，分别反映所发现的规则的有用性和确定性；
  - 如果关联规则同时满足最小支持度阈值(min_sup)和最小置信度阈值(min_conf)，那么称它是有趣的；
  - 最小支持度阈值和最小置信度阈值可以由相关领域专家设定；

关联规则基本概念

- 设I是项的集合，关联规则：A → B 满足A，B是 I 的子集，且A∩B = 空集；
- 支持度
  - support(A → B) = P(A∪B)；事务中同时包含X和Y的概率；
- 置信度
  - confidence(A → B) = P(B|A)；事务中包含A的情况下也包括B的条件概率；
- 强关联规则
  - 同时满足最小支持度阈值（min_sup）和最小置信度阈值（min_conf）的规则称作强规则。
  - 例：computer → antivirus_software[support=2%, confidence=60%]

频繁项集基本概念

- 项集：项的集合；
  - k-项集：包含k个项的项集；
  - 项集的支持度：项集出现的频率（包含项集的事务数）
  - 频繁项集：支持度不小于预定义的最小支持度阈值的项集；频繁k-项集通常记做L_k；



频繁项集和关联规则

- 关联规则挖掘过程
  - 找出所有频繁项集
    - 所有项集满足最小支持度阈值
  - 由频繁项集产生关联规则
    - 满足最小支持度和最小置信度

由频繁项集$l=\{I1,I2,I5\}$可以产生那些关联规则？

L的非空子集有$\{I1,I2\},\{I1,I5\},\{I2,I5\},\{I1\},\{I2\},\{I5\}$，则产生如下关联规则：

I1ÙI2 Þ I5　　I1ÙI5 Þ I2　　I2ÙI5 Þ I1

I1 Þ I2ÙI5　　I2 Þ I1ÙI5　　I5 Þ I1ÙI2

提纲

1. 基本概念
2. 频繁项集挖掘方法
3. 哪些模式是有趣的：模式评估方法
4. 小结

有效的和可伸缩的频繁项集挖掘方法

- Apriori算法：使用候选产生发现频繁项集
  - 由频繁项集产生关联规则
- 提高Apriori算法的效率
- 不候选产生挖掘频繁项集
- 使用垂直数据格式挖掘频繁项集
- 挖掘闭频繁项集

Apriori算法

- Apriori算法
  - 美国学者 Rakesh Agrawal提出
  - 1994年发表于VLDB国际会议

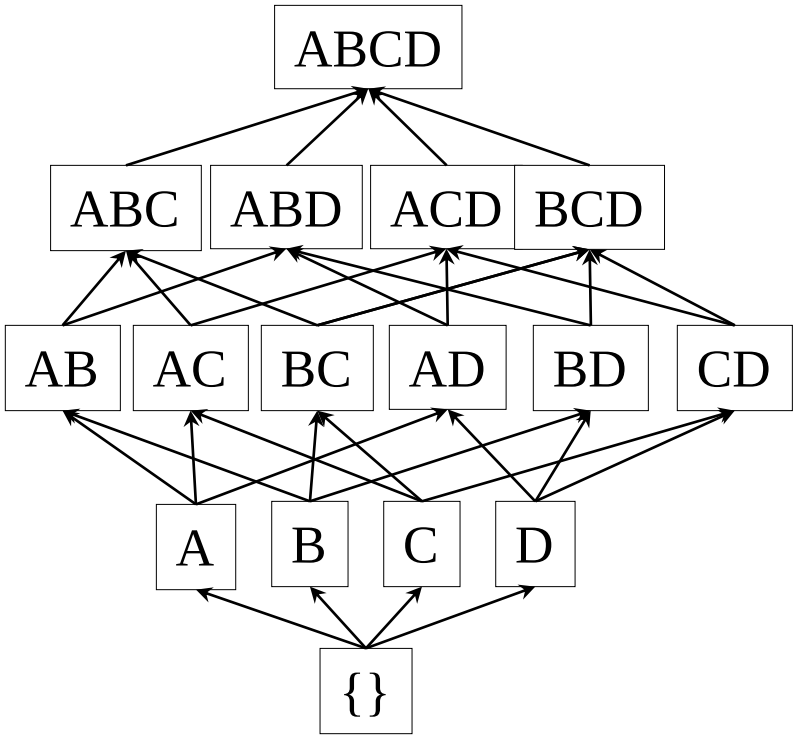
Fast Algorithms for Mining Association Rules

Apriori算法在ICDM'06被评为十大数据挖掘算法第四名

- **#1: C4.5 (61 votes)**
- **#2: K-Means (60 votes)**
- **#3: SVM (58 votes)**
- **#4: Apriori (52 votes)**
- **#5: EM (48 votes)**
- **#6: PageRank (46 votes)**
- **#7: AdaBoost (45 votes)**
- **#7: kNN (45 votes)**
- **#7: Naive Bayes (45 votes)**
- **#10: CART (34 votes)**

Apriori算法

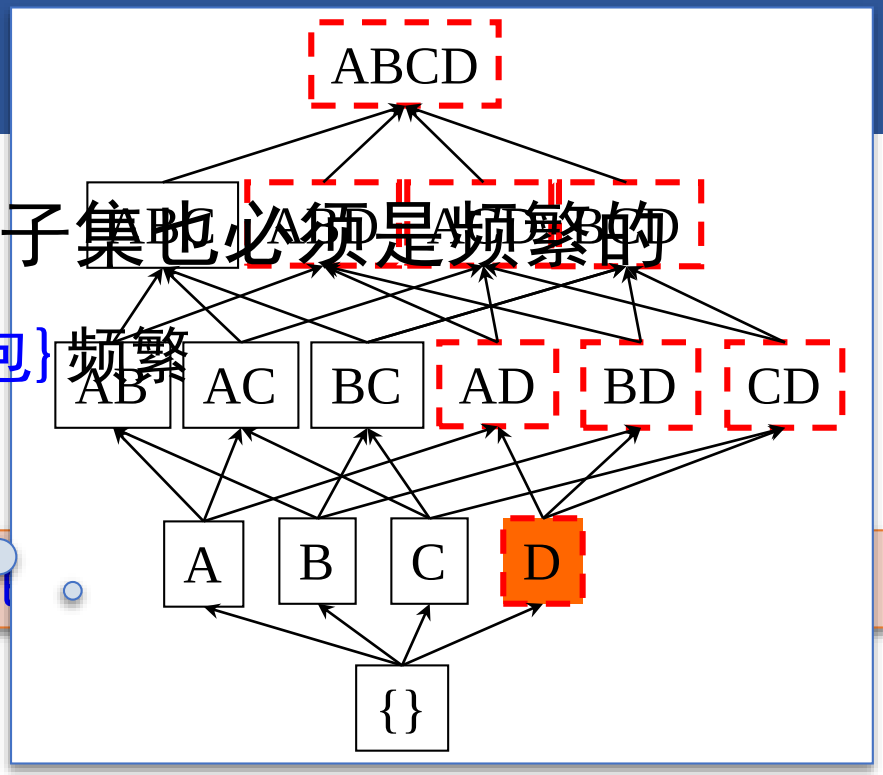
- Apriori算法思想
  - 逐层搜索的迭代方法
  - 用k-项集探索(k+1)-项集
    - 扫描数据库，发现L1
    - $L_1 \rightarrow L_2$
    - $L_2 \rightarrow L_3$
    - ……
    - $L_k \rightarrow L_{k+1}$



Apriori算法

- Apriori性质：频繁项集的所有非空子集也必须是频繁的
  - {牛奶，面包，糖}频繁→{牛奶，面包}频繁

$L_k \rightarrow L_{k+1}$



min_sup ≦ sup({牛奶，面

- Apriori剪枝规则
  - 如果一个项集是非频繁的，则它的所有超集也都是非频繁的；
  - {牛奶，面包}非频繁→{牛奶，面包，糖}非频繁

$$min\_sup \geqq sup(\{牛奶，面包\}) \geqq sup(\{牛奶，面包，糖\})$$

Apriori算法

- 如何用L_k找L_{k+1}：
- Step1: 连接（Self-joining）
  - $L_k * L_k$自连接产生候选(k+1)项集的集合C_{k+1}。
  - 设$l_1, l_2 \in L_k$, 如果

$(l_1[1]=l_2[1]) \wedge \ldots \wedge (l_1[k-1]=l_2[k-1]) \wedge (l_1[k]<l_2[k])$

则连接结果为

$C_{k+1}=\{l_1[1], l_1[2], \ldots, l_1[k-1], l_1[k], l_2[k]\}$

- Step2:剪枝（Pruning）
  - 压缩C_{k+1}，删除包含非频繁k项子集的(k+1)项集
  - 依次确定C_{k+1}中每个项集的支持度计数，从而确定L_{k+1}

Apriori算法

- 例:连接与剪枝
  - L_3={{abc}, {abd}, {acd}, {ace}, {bcd}}
  - 自连接: L_3*L_3
    - {abcd} ← {abc} & {abd}
    - {acde} ← {acd} & {ace}
  - 剪枝
    - {acde}的子集{ade}不属于L_3(非频繁的)，因此去除{acde}
    - C_4={{abcd}}

Apriori算法

事务数据集

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

C_1

1st scan

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

min_sup= 2

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2nd scan

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

Itemset
{B, C, E}

3rd scan

L_3

Itemset	sup
{B, C, E}	2

Apriori算法

输入：事务数据库 D；最小支持度阈值。

输出：D 中的频繁项集 L。

方法：

1) L_1 = find_frequent_1_itemsets(D);
2) **for** (k = 2; $L_{k-1} \neq \varnothing$; k++) {
3) C_k = aproiri_gen(L_{k-1},min_sup);
4) **for each** transaction $t \in D${
5) C_t = subset(C_k,t);
6) **for each** candidate $c \in C_t$
7) c.count++;
8) }
9) L_k={$c \in C_k$ | c.count ≥ min_sup}
10) }
11) **return** $L = \cup_k L_k$;

迭代

$L_k \rightarrow C_{k+1}$

procedure apriori_gen(L_{k-1}: frequent (k-1)-itemset; min_sup: supp
1) **for each** itemset $l_1 \in L_{k-1}$
2) **for each** itemset $l_2 \in L_{k-1}$
3) **if** ($l_1[1]=l_2[1] \wedge ... \wedge (l_1[k-2]=l_2[k-2]) \wedge (l_1[k-1]<l_2[k-2])$ **then** {
4) c = l_1 l_2; //join step: generate candidates
5) **if** has_infrequent_subset(c,L_{k-1}) **then**
6) **delete** c; // prune step: remove unfrequent cadidate
7) **else add** c **to** C_k;
8) }
9) **return** C_k;

连接

剪枝

由频繁项集产生关联规则

- 强关联规则满足最小支持度和最小置信度

$$\text{confidence}(A \Rightarrow B) = \frac{\text{support\_count}(A \cup B)}{\text{support\_count}(A)}$$

  - 其中support_count(A∪B) 是包含项集A∪B的事务数， support_count(A)是包含项A的事务数。

- 产生方法:
  - 对每个频繁项集*l*，产生*l*的所有非空子集
  - 对*l*的每个非空子集s，如果 $\frac{\text{support\_count}(l)}{\text{support\_count}(s)} \geq \text{min\_conf}$
  - 则输出规则“s → (*l* - s)”，
  - 其中： min_conf 是最小置信度

由频繁项集产生关联规则

事务数据集

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

L_3

Itemset	sup
{B, C, E}	2

由频繁项集 $l=\{B,C,E\}$ 可以产生哪些关联规则？l 的非空子集有 $\{B,C\},\{B,E\},\{C,E\},\{B\},\{C\},\{E\}$,则产生如下关联规则：

$B \wedge C \Rightarrow E$ confidence = 2/2 = 100%　　$B \wedge E \Rightarrow C$ confidence = 2/3 = 67%

$C \wedge E \Rightarrow B$ confidence = 2/2 = 100%　　$B \Rightarrow C \wedge E$ confidence = 2/3 = 33%

$C \Rightarrow B \wedge E$ confidence = 2/3 = 67%　　$E \Rightarrow B \wedge C$ confidence = 2/3 = 67%

如果最小置信度阈值为70%，则只有2条强规则可以输出。

Apriori缺陷

- Apriori算法的核心:
  - 用频繁的(k–1)-项集生成候选的k-项集
  - 用数据库扫描和模式匹配计算候选集的支持度

- Apriori 的瓶颈: 候选集生成
  - 巨大的候选集:
    - 10^4个频繁1-项集要生成10^7个候选2-项集
  - 多次扫描数据库
    - 如果最长的模式是n的话，则需要(n+1)次数据库扫描

有效的和可伸缩的频繁项集挖掘方法

- Apriori算法：
  - 使用候选产生发现频繁项集
  - 由频繁项集产生关联规则
- 提高Apriori算法的效率
- 不候选产生挖掘频繁项集
- 使用垂直数据格式挖掘频繁项集
- 挖掘闭频繁项集

几种提高Apriori效率的方法

- 基于散列的技术
- 划分方法
- 事务压缩
- 抽样
- 动态项计数

提高Apriori算法的效率

- 基于散列的技术（散列项集到对应的桶中）
  - 例：统计C_1的支持度计数时，对每个事务产生的2项集，将其散列到散列表的桶中，并增加桶的计数；对于计数小于最小支持度的桶，可以从候选集中删除。

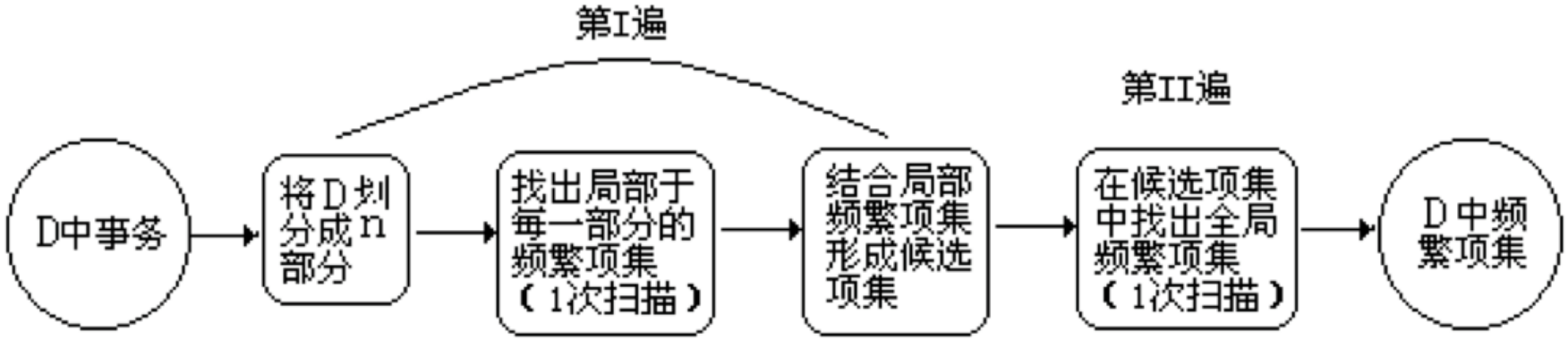
TID	项ID的列表
T100	I1,I2,I5
T200	I2,I4
T300	I2,I3
T400	I1,I2,I4
T500	I1,I3
T600	I2,I3
T700	I1,I3
T800	I1,I2,I3,I5
T900	I1,I2,I3

桶地址	0	1	2	3	4	5	6
桶计数	2	2	4	2	2	4	4
桶内容	{I1,I4} {I3,I5}	{I1,I5} {I1,I5}	{I2,I3} {I2,I3} {I2,I3} {I2,I3}	{I2,I4} {I2,I4}	{I2,I5} {I2,I5}	{I1,I2} {I1,I2} {I1,I2} {I1,I2}	

$$h(I_i, I_j) = (i' \times 10 + j' \times 7) \bmod 7$$

提高Apriori算法的效率

- 划分（为寻找候选项集划分数据）
  - 阶段1：将D中的事务分成n个非重叠的划分，找出该划分内的所有频繁项集。
  - 阶段2：第二次扫描，评估每个候选的实际支持度，以确定全局频繁项集。



提高Apriori算法的效率

- 事务压缩（压缩未来迭代扫描的事务数）
  - 不包含任何频繁k-项集的事务不可能包含任何(k+1)-项集。
  - 可以加上标签或删除，因为产生j项集（j>k）的数据库扫描不再需要它们。
- 抽样（对给定数据的子集挖掘）
  - 基本思想：选取给定数据D的随机样本S，然后在S中搜索频繁项集。
  - 牺牲精度换取效率。
- 动态项集计数（在扫描的不同点添加候选项集）
  - 可以在任何时候添加候选项集。

提纲

1. 基本概念
2. 频繁项集挖掘方法(续)
3. 哪些模式是有趣的：模式评估方法
4. 小结