Pattern Recognition & Machine Learning

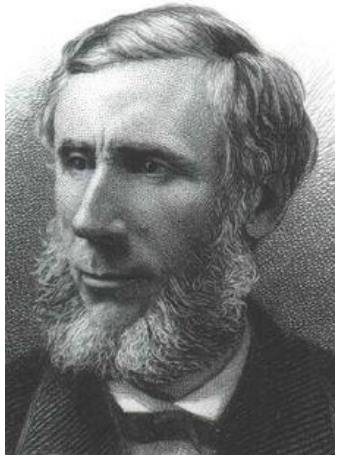
80206085244011（40/2）

Association Rules

Wang Yong

University of Chinese Academy of Sciences

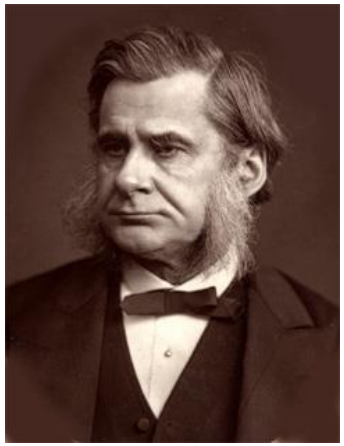
2018.04.22



有了精确的实验和观测作为研究的依据，想象力便成为自然科学理论的设计师。

——John Tyndall

英国物理学家 (1820 – 1893)



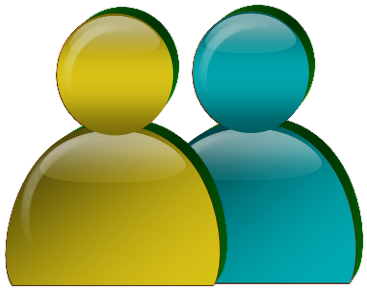
我要做的是叫我的愿望符合事实，而不是试图让事实与我的愿望调和。你们要像一个小学生那样坐在事实面前，准备放弃一切先人之见，恭恭敬敬地照着大自然指的路走，否则，就将一无所有。

—— Thomas Henry Huxley

英国生物学家 (1825 – 1895)

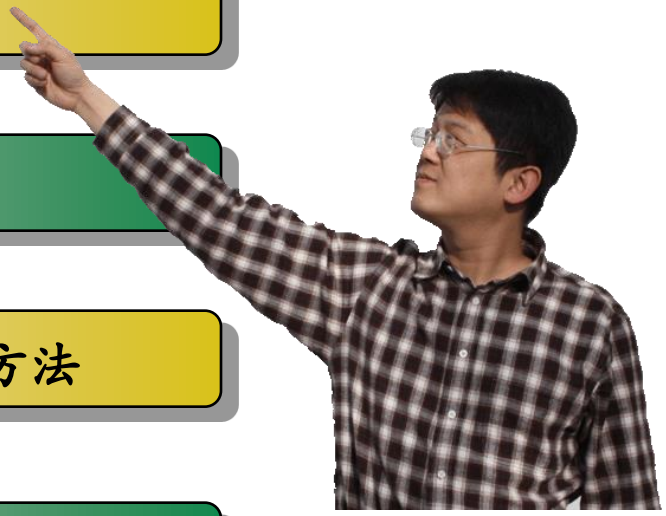
关联规则模式发现与识别

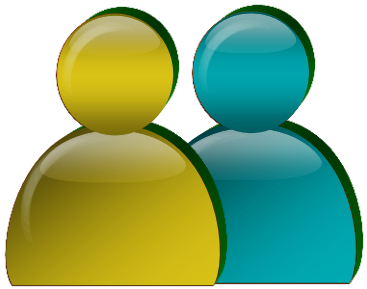
1. 基本概念
2. Apriori算法
3. 发现频繁项集的其他方法
4. 关联规则的扩展



关联规则模式发现与识别

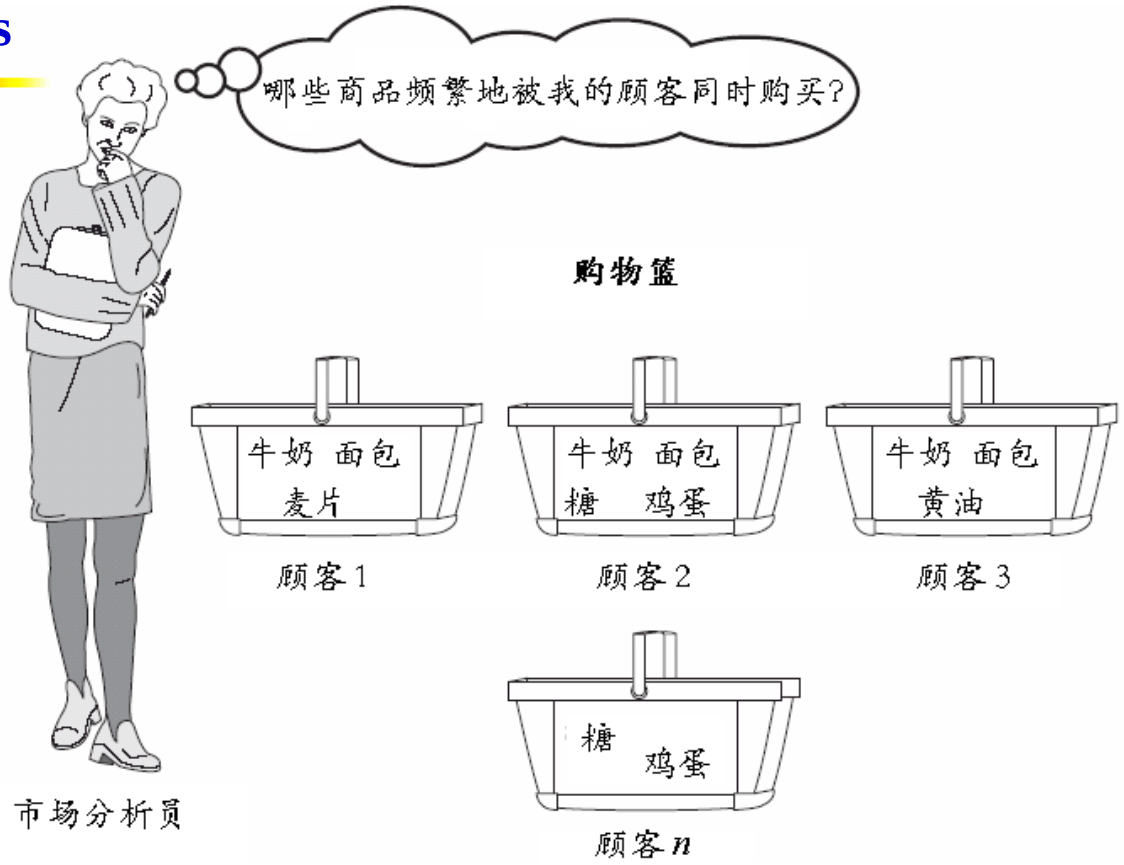
1. 基本概念
2. Apriori算法
3. 发现频繁项集的其他方法
4. 关联规则的扩展





关联规则模式发现与识别

Frequent Patterns



关联规则发现是发现事务或关系数据库中项集之间有趣的关联或相关的**频繁模式**，更高级的频繁模式发现包括序列模式发现（时间序列、空间序列、逻辑序列）和结构模式发现（子图、子树、子格）

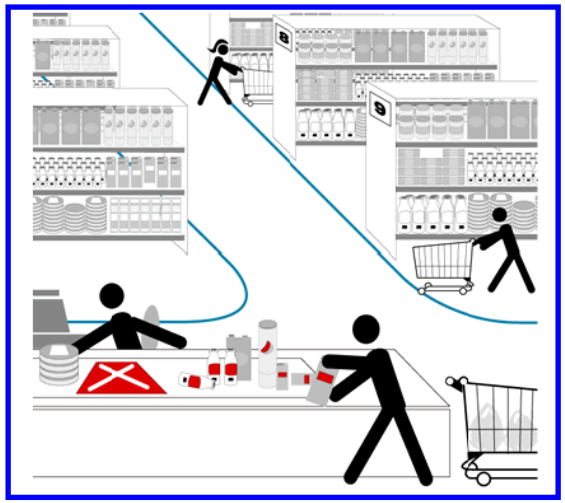
关联规则是描述两个或多个变量之间的某种潜在关系的特征规则

基本概念

- 简言之，关联规则模式是研究“什么与什么相伴”，最早由Agrawal等人于1993年首次提出。

R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large Databases. In Proc. 1993 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'93), pages 207-216, Washington, DC, May 1993.

- 关联是指两个或多个变量的取值之间存在某种规律性
- 关联规则关注事务项之间的关联性，而不考虑数量
- 关联规则模式最初提出的动机是针对购物篮分析问题，目的是从交易数据库中发现顾客购物的行为规则。



样本数据集 D_{sample}

事务	事务标识（tid）	项目
t_1	1	面包，牛奶
t_2	2	面包，尿布，啤酒，鸡蛋
t_3	3	牛奶，尿布，啤酒，可乐
t_4	4	面包，牛奶，尿布，啤酒
t_5	5	面包，牛奶，尿布，可乐

D_{sample} 的项目全集为 I = {面包，牛奶，尿布，啤酒，鸡蛋，可乐}

基本概念

- **定义1：事务（transaction）**
  - 某个客户在一次交易中发生的所有项目的集合
  - 组成事务的项目集合是 I 上的一个子集
  - 每个事务都有一个唯一的标识的 tid

样本数据集 D_{sample}

事务	事务标识（tid）	项目
t_1	1	面包，牛奶
t_2	2	面包，尿布，啤酒，鸡蛋
t_3	3	牛奶，尿布，啤酒，可乐
t_4	4	面包，牛奶，尿布，啤酒
t_5	5	面包，牛奶，尿布，可乐

I = {面包，牛奶，尿布，啤酒，鸡蛋，可乐}

- **定义2：事务数据库（transaction database）$D=\{t_1, t_2, \ldots, t_n\}$**
  - 由一系列具有唯一标识 tid 的事务组成

- **定义3：项（item）**
  - 事务数据库中的一个属性字段，每个字段有一定的取值范围
  - 对于超市数据来说，项是指交易中的特定商品

- **定义4：项集（itemset）**
  - 包含若干个项的集合

项集人构造
事务不可变

基本概念

■ **定义5：项集维数**

- 把一个项集所包含的项的个数称为此项集的维数或项集的长度
- 长度为 *k* 的项集，称为 *k* - 项集

样本数据集 D_{sample}

事务	事务标识（tid）	项目
t_1	1	面包，牛奶
t_2	2	面包，尿布，啤酒，鸡蛋
t_3	3	牛奶，尿布，啤酒，可乐
t_4	4	面包，牛奶，尿布，啤酒
t_5	5	面包，牛奶，尿布，可乐

I = {面包，牛奶，尿布，啤酒，鸡蛋，可乐}

■ **定义6：支持度（support）**

- 假定 *X* 是一个项集，*D* 是一个事务集合或事务数据库，称 *D* 中包含 *X* 的交易的个数与 *D* 中总的交易个数之比为 *X* 在 *D* 中的支持度，记作 Support（*X*），即

$$\text{Support } X = \frac{\| d \in D \mid X \in d \|}{\|D\|}$$

基本概念

- 示例

样本数据集 D_{sample}

事务	事务标识（tid）	项目
t_1	1	面包，牛奶
t_2	2	面包，尿布，啤酒，鸡蛋
t_3	3	牛奶，尿布，啤酒，可乐
t_4	4	面包，牛奶，尿布，啤酒
t_5	5	面包，牛奶，尿布，可乐

- 样本数据集D_{sample} = {t_1，t_2，t_3，t_4，t_5} 被称为事务数据库；交易记录t_1代表了一个事务，它表示在某次购物中购买了“面包”和“牛奶”，“面包”和“牛奶”就代表了 2 个不同的项。
- {面包、牛奶} 、 {面包} 、 {面包、啤酒} 等都是项集，但只有 {面包、牛奶} 才是一个事务。项集 {面包、牛奶} 的维数为 2，记作 2 - 项集。
- 假定项集 X = {面包、牛奶} ，则在 D_{sample} 中包含项集 X 的有t_1，t_4，t_5；因为 D_{sample} 中的事务个数为 5，故 X 在 D_{sample} 中的支持度为 60%。

基本概念

样本数据集 D_{sample}

事务	事务标识（tid）	项目
t_1	1	面包，牛奶
t_2	2	面包，尿布，啤酒，鸡蛋
t_3	3	牛奶，尿布，啤酒，可乐
t_4	4	面包，牛奶，尿布，啤酒
t_5	5	面包，牛奶，尿布，可乐

所有项目全集的支持度

项集	支持度%	项集	支持度%	项集	支持度%
面包	80	牛奶，鸡蛋	0	面包，尿布，鸡蛋	20
牛奶	80	牛奶，可乐	40	面包，尿布，可乐	20
尿布	80	尿布，啤酒	60	面包，啤酒，鸡蛋	20
啤酒	60	尿布，鸡蛋	20	面包，啤酒，可乐	0
鸡蛋	20	尿布，可乐	40	面包，鸡蛋，可乐	0
可乐	40	啤酒，鸡蛋	20	牛奶，尿布，啤酒	40
面包，牛奶	60	啤酒，可乐	20	牛奶，尿布，鸡蛋	0
面包，尿布	60	鸡蛋，可乐	0	牛奶，尿布，可乐	40
面包，啤酒	40	面包，牛奶，尿布	40	牛奶，啤酒，鸡蛋	0
面包，鸡蛋	20	面包，牛奶，啤酒	20	牛奶，啤酒，可乐	20
面包，可乐	20	面包，牛奶，鸡蛋	0	牛奶，鸡蛋，可乐	0
牛奶，尿布	60	面包，牛奶，可乐	20	尿布，啤酒，鸡蛋	20
牛奶，啤酒	40	面包，尿布，啤酒	40	尿布，啤酒，可乐	20

所有项目全集的支持度（续）

项集	支持度%	项集	支持度%
尿布，鸡蛋，可乐	0	牛奶，尿布，啤酒，可乐	20
啤酒，鸡蛋，可乐	0	牛奶，尿布，鸡蛋，可乐	0
面包，牛奶，尿布，啤酒	20	牛奶，啤酒，鸡蛋，可乐	0
面包，牛奶，尿布，鸡蛋	0	尿布，啤酒，鸡蛋，可乐	0
面包，牛奶，尿布，可乐	20	面包，牛奶，尿布，啤酒，鸡蛋	0
面包，牛奶，啤酒，鸡蛋	0	面包，牛奶，尿布，啤酒，可乐	0
面包，牛奶，啤酒，可乐	0	面包，牛奶，尿布，鸡蛋，可乐	0
面包，牛奶，鸡蛋，可乐	0	面包，牛奶，啤酒，鸡蛋，可乐	0
面包，尿布，啤酒，鸡蛋	20	面包，尿布，啤酒，鸡蛋，可乐	0
面包，尿布，啤酒，可乐	0	牛奶，尿布，啤酒，鸡蛋，可乐	0
面包，尿布，鸡蛋，可乐	0	面包，牛奶，尿布，啤酒，鸡蛋，可乐	0
面包，啤酒，鸡蛋，可乐	0		
牛奶，尿布，啤酒，鸡蛋	0		

基本概念

- **定义7：最小支持度（minimum support）**
  - 由用户定义的衡量项集频繁程度的一个阈值，记作minsup
- **定义8：频繁项集（frequent itemset）**
  - 对一个项集 X，如果 X 的支持度不小于最小的支持度，即Support（X）≥ minsup，则称 X 为频繁项集或大项集（large itemset）
- **定义9：非频繁项集（non - frequent itemset）**
  - 对一个项集 X，如果 X 的支持度小于最小的支持度，即Support（X）≤ minsup，则称 X 为非频繁项集或小项集（small itemset）
- **定义10：最大频繁项集（maximum frequent itemset）**
  - 在频繁项集中，挑选出所有不被其他元素包含的频繁项集，称为最大频繁项集或最大大项集（maximum large itemset）

样本数据集 D_{sample}

事务	事务标识（tid）	项目
t_1	1	面包，牛奶
t_2	2	面包，尿布，啤酒，鸡蛋
t_3	3	牛奶，尿布，啤酒，可乐
t_4	4	面包，牛奶，尿布，啤酒
t_5	5	面包，牛奶，尿布，可乐

I = {面包，牛奶，尿布，啤酒，鸡蛋，可乐}

基本概念

- **示例**

样本数据集 D_{sample}

事务	事务标识（tid）	项目
t_1	1	面包，牛奶
t_2	2	面包，尿布，啤酒，鸡蛋
t_3	3	牛奶，尿布，啤酒，可乐
t_4	4	面包，牛奶，尿布，啤酒
t_5	5	面包，牛奶，尿布，可乐

- 如果最小支持度设置为60%，则例题中，所有的频繁项集和最大频繁项集为：

项集	支持度%
面包	80
牛奶	80
尿布	80
啤酒	60

项集	支持度%
面包，牛奶	60
面包，尿布	60
牛奶，尿布	60
尿布，啤酒	60

项集	支持度%
面包，牛奶	60
面包，尿布	60
牛奶，尿布	60
尿布，啤酒	60

基本概念

- **定义 11：置信度（confidence）**

- 对形如 $X \rightarrow Y$ 的关联规则（X 和 Y 都是项集），定义规则的置信度为事务集合 D 中既包含 X 也包含 Y 的事务个数与 D 中包含 X 的事务个数之比，或者说是项集 $X \cup Y$ 的支持度与 X 的支持度之比，记作

$$\text{Confidence}\ X \rightarrow Y = \frac{\text{Support}\ X \cup Y}{\text{Support}\ X}$$

- **定义 12：最小置信度（minimum confidence）**

- 用户定义的一个置信度阈值，表示规则的最低可靠性，记作**minconf**

- **为什么使用支持度和置信度?**

- 支持度很低的规则可能只是偶然出现，支持度通常用来删除那些不令人感兴趣的规则

- 置信度衡量通过规则进行推理的可靠性，置信度越高，规则越可靠

基本概念

- 示例

样本数据集 D_{sample}

事务	事务标识（tid）	项目
t_1	1	面包，牛奶
t_2	2	面包，尿布，啤酒，鸡蛋
t_3	3	牛奶，尿布，啤酒，可乐
t_4	4	面包，牛奶，尿布，啤酒
t_5	5	面包，牛奶，尿布，可乐

项集	支持度%	项集	支持度%
面包	80	面包，牛奶	60
牛奶	80	面包，尿布	60
尿布	80	牛奶，尿布	60
啤酒	60	尿布，啤酒	60

- 如果最小支持度设置为60%，则例题中，可生成的规则为：

$X \rightarrow Y$	支持度%	置信度%	$X \rightarrow Y$	支持度%	置信度%
面包→牛奶	60	75	牛奶→面包	60	75
面包→尿布	60	75	尿布→面包	60	75
牛奶→尿布	60	75	尿布→牛奶	60	75
尿布→啤酒	60	75	啤酒→尿布	60	100

基本概念

- 示例

样本数据集 *D*sample

事务	事务标识（tid）	项目
t_1	1	牛奶，面包
t_2	2	面包，黄油
t_3	3	牛奶，面包，黄油
t_4	4	牛奶

牛奶	面包	黄油
1	1	0
0	1	1
1	1	1
1	0	0

- 从以上三件商品中选出两件进行搭配销售，哪个搭配销售方案更好?

牛奶+面包	牛奶+黄油	面包+黄油
1/2 ; 2/3	1/4 ; 1/3	1/2 ; 2/3
面包+牛奶	黄油+牛奶	黄油+面包
1/2 ; 2/3	1/4 ; 1/2	1/2 ; 1

关联规则模式发现与识别

- 关联规则发现的基本任务就是通过用户指定最小支持度（minsup）和最小置信度（minconf），发现大型数据库中的强关联规则

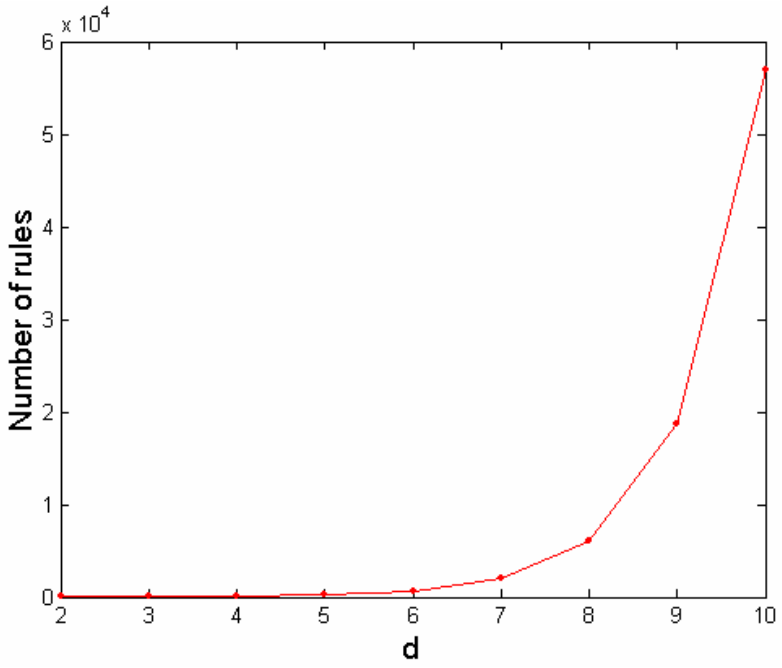
（1）频繁项集的产生：发现满足最小支持度阈值的所有项集，这些项集被称作频繁项集（frequent itemset）

（2）规则的产生：从上一步发现的频繁项集中提取所有高置信度的规则，这些规则被称作强规则（strong rule）

- 关联规则模式发现的一种原始方法：

  计算每个可能规则的支持度和置信度

计算代价太高！



- 一个包含 d 个项的数据集可以产生 2^d-1 个频繁项集
- 可以产生 $3^d-2^{d+1}+1$ 条规则

关联规则模式发现与识别

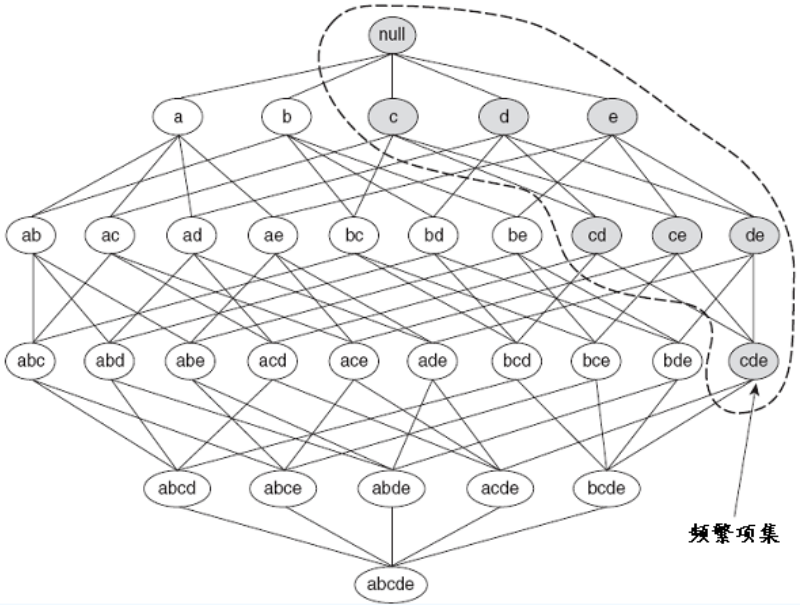
- 如何降低产生频繁项集的计算复杂度?

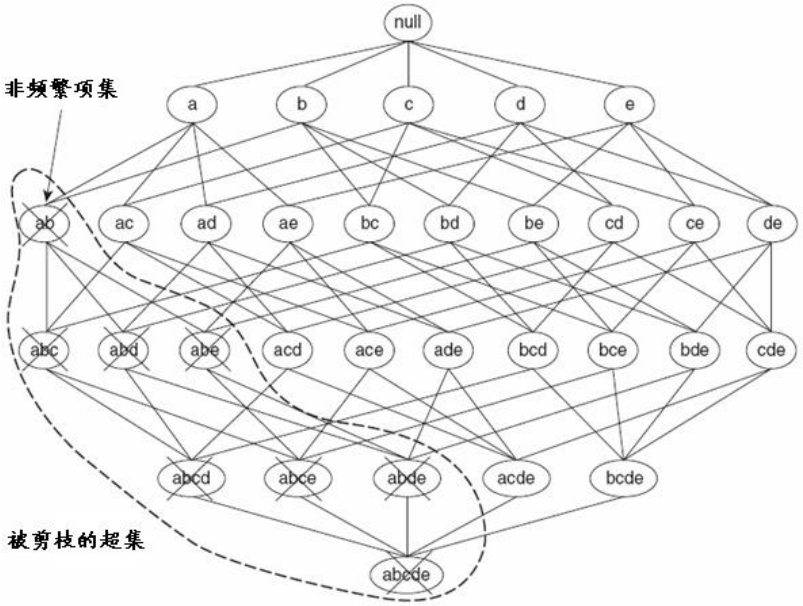
① 减少候选项集的数目（Reduce the number of candidates）：

不用计算支持度值就直接删除某些候选项集

- 先验原理：如果一个项集是频繁的，则它的所有子集也一定是频繁的

  如果一个项集是非频繁的，则它的所有超集也一定是非频繁的





- 支持度度量的反单调性（anti-monotone）

一个项集的支持度绝不会超过它的子集的支持度

$$\forall X, Y \in J:$$

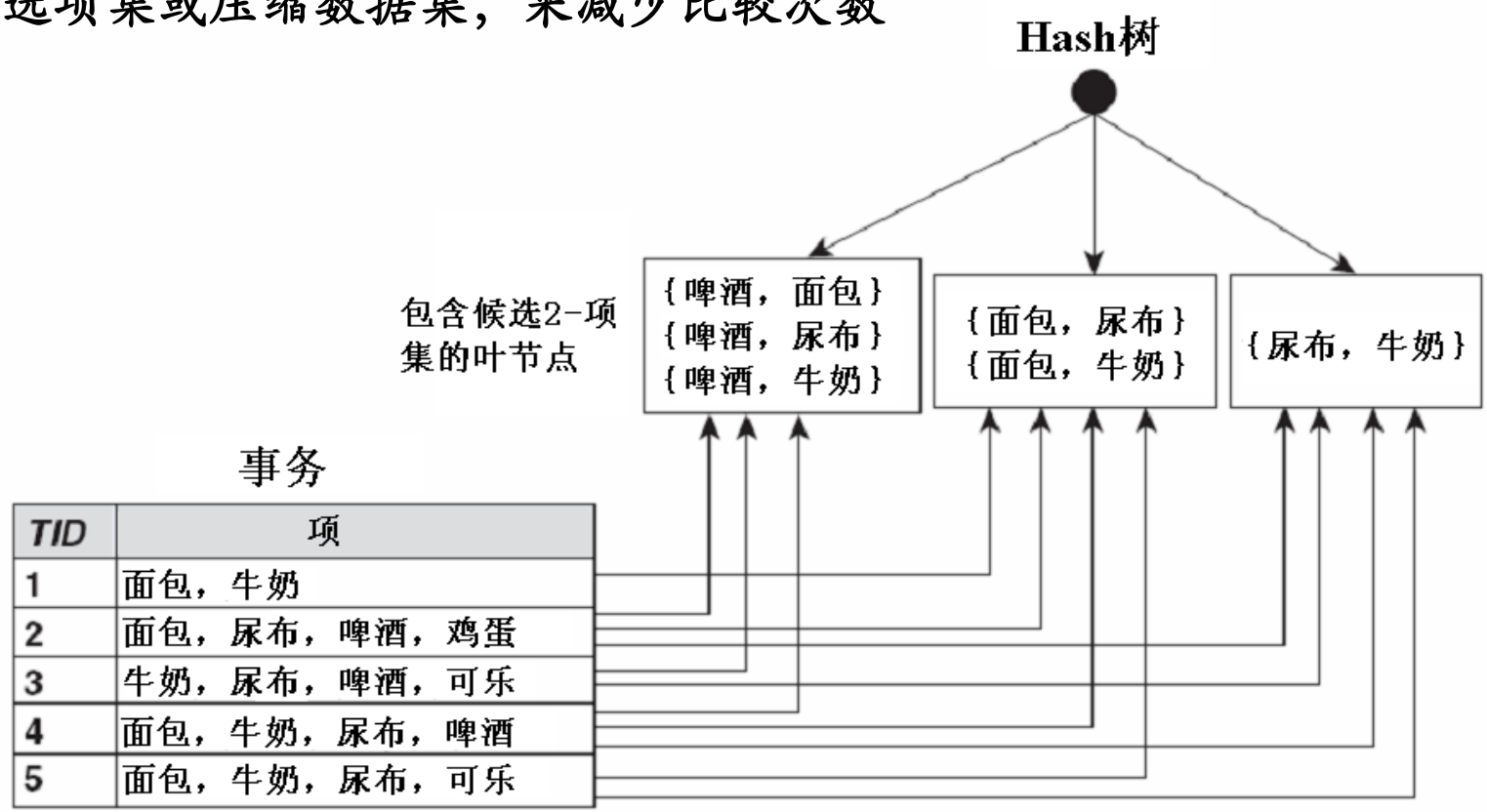
$$X \subseteq Y \rightarrow s\ X \geq s\ Y$$

关联规则模式发现与识别

- 如何降低产生频繁项集的计算复杂度?

② 计算支持度时减少比较次数（Reduce the number of comparisons）：

不用将每个候选项集与每个事务进行匹配，通过使用更高级的数据结构存储候选项集或压缩数据集，来减少比较次数



事务

TID	项
1	面包，牛奶
2	面包，尿布，啤酒，鸡蛋
3	牛奶，尿布，啤酒，可乐
4	面包，牛奶，尿布，啤酒
5	面包，牛奶，尿布，可乐

关联规则模式发现与识别

- 如何降低产生频繁项集的计算复杂度?

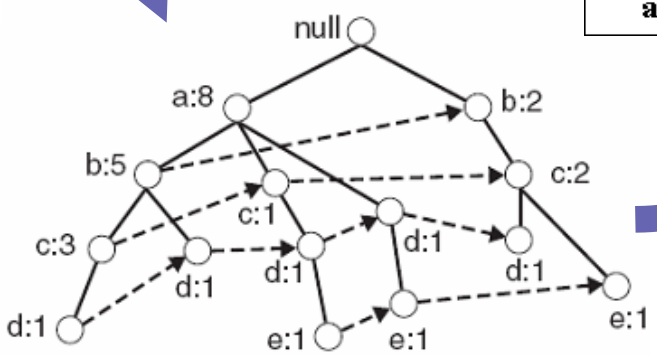
③ 直接产生频繁项集（频繁模式增长; frequent-pattern growth; FP增长）

- 将提供频繁项的数据库压缩到一棵频繁模式树（或 FP树），但仍保留项集关联信息
- 将压缩后的数据库分成一组条件数据库（一种特殊类型的投影数据库），每个关联一个频繁项或“模式段”，并分别挖掘每个条件数据库

事务数据集

TID	项
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}

构造FP树

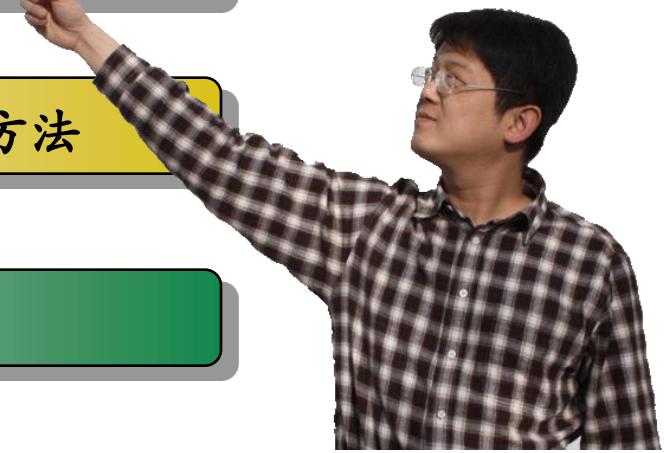


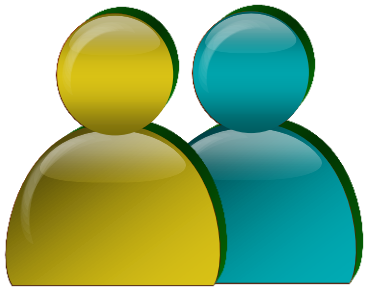
依据相应的后缀排序的频繁

后 缀	频 繁 项 集
e	{e}, {d,e}, {a,d,e}, {c,e}, {a,e}
d	{d}, {c,d}, {b,c,d}, {a,c,d}, {b,d}, {a,b,d}, {a,d}
c	{c}, {b,c}, {a,b,c}, {a,c}
b	{b}, {a,b}
a	{a}

关联规则模式发现与识别

1. 基本概念
2. Apriori算法
3. 发现频繁项集的其他方法
4. 关联规则的扩展





Apriori算法

- **Apriori算法是第一个关联规则模式发现算法**
  - R.Agrawal 和 R.Srikant 于1994年提出的为布尔关联规则发现频繁项集的原创性算法
  - 开创性地使用基于支持度的剪枝技术（先验原理），使用逐层搜索（宽度优先）的迭代搜索方法，系统地控制候选项集指数增长
- **Apriori算法的基本思想**
  - 首先，通过扫描数据库，累计每个项的计数，并收集满足最小支持度的项，找出频繁 1 项集的集合，该集合记作 L_1
  - 然后，L_1用于找频繁 2 项集的集合 L_2
  - L_2 用于找频繁 3 项集的集合 L_3
  - 如此下去，直到不能再找到频繁 k 项集

找每个 L_k 都需要一次数据库全扫描

Apriori算法

- 示例

样本数据集 D_{sample}

事务	事务标识（tid）	项目
t_1	1	面包，牛奶
t_2	2	面包，尿布，啤酒，鸡蛋
t_3	3	牛奶，尿布，啤酒，可乐
t_4	4	面包，牛奶，尿布，啤酒
t_5	5	面包，牛奶，尿布，可乐

假定支持度阈值是60%，相当于最小支持度计数为3

枚举所有项集（到 3 项集）的蛮力策略将产生

$C_6^1 + C_6^2 + C_6^3 = 41$ 个候选

使用先验原理，将减少为

$C_6^1 + C_4^2 + 1 = 13$ 个候选

候选 1 项集

项 集	计 数
啤 酒	3
面 包	4
可 乐	2
尿 布	4
牛 奶	4
鸡 蛋	1

因支持度低而被删除的项集

候选 2 项集

项 集	计 数
{啤酒，面包}	2
{啤酒，尿布}	3
{啤酒，牛奶}	2
{面包，尿布}	3
{面包，牛奶}	3
{尿布，牛奶}	3

候选 3 项集

项 集	计 数
{面包，尿布，牛奶}	3

Apriori算法

■ Apriori算法的频繁项集产生

1: $k = 1$

2: $F_k = \{ i \mid i \in I \wedge \sigma(\{i\}) \geq N \times minsup \}$ {发现所有的频繁 1 项集}

3: **repeat**

4: $k = k + 1$

5: $C_k = \text{apriori\_gen}(F_{k-1})$ {产生候选项集}

6: **for** 每个事务 $t \in T$ **do**

7: $C_t = \text{subset}(C_k, t)$ {识别属于 t 的所有候选}

8: **for** 每个候选项集 $c \in C_t$ **do**

9: $\sigma(c) = \sigma(c) + 1$ {支持度计数增值}

10: **end for**

11: **end for**

12: $F_k = \{ c \mid c \in C_k \wedge \sigma(\{c\}) \geq N \times minsup \}$ {提取频繁 k 项集}

13: **until** $F_k = \varnothing$

14: Result = $\cup F_k$

候选项集的产生

扫描全部数据

候选项集的剪枝

Apriori算法

■ 候选项集的产生 Procedure apriori_gen (F_{k-1})

① 蛮力方法：把所有的 k 项集都看作可能的候选，然后使用候选剪枝除去不必要的候选

项

项
啤酒
面包
可乐
尿布
牛奶
鸡蛋

候选产生

项集
{啤酒，面包，可乐}
{啤酒，面包，尿布}
{啤酒，面包，牛奶}
{啤酒，面包，鸡蛋}
{啤酒，可乐，尿布}
{啤酒，可乐，牛奶}
{啤酒，可乐，鸡蛋}
{啤酒，尿布，牛奶}
{啤酒，尿布，鸡蛋}
{啤酒，牛奶，鸡蛋}
{面包，可乐，尿布}
{面包，可乐，牛奶}
{面包，可乐，鸡蛋}
{面包，尿布，牛奶}
{面包，尿布，鸡蛋}
{面包，牛奶，鸡蛋}
{可乐，尿布，牛奶}
{可乐，尿布，鸡蛋}
{可乐，牛奶，鸡蛋}
{尿布，牛奶，鸡蛋}

候选产生相当简单，候选剪枝开销极大

第 k 产生的候选项集数目为 C_d^k

（d 是项的总数}

候选剪枝

项集
{面包，尿布，牛奶}

设每个候选项集所需的计算量为 $O(k)$

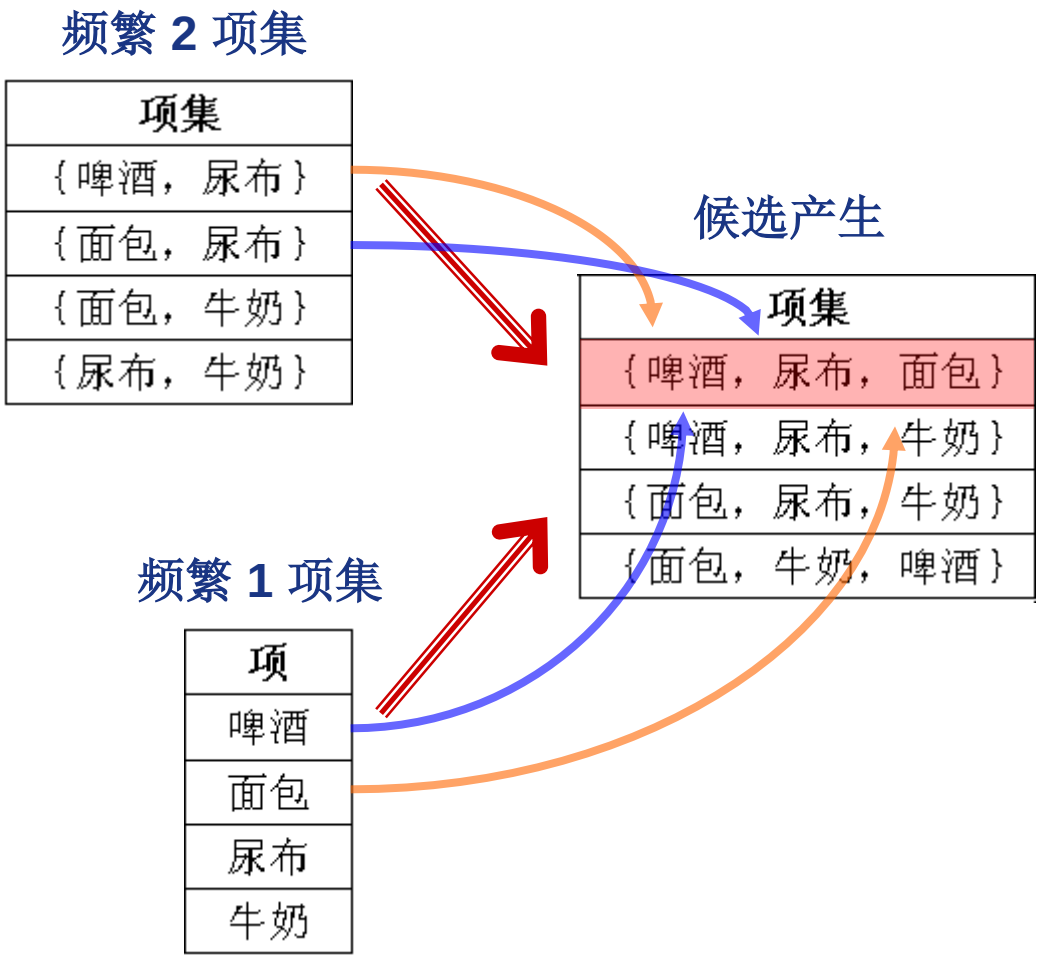
蛮力方法的总复杂度为 $O(\Sigma_{k=1}^{d} kC_d^k)$

$= O(d \cdot 2^{d-1})$

Apriori算法

■ 候选项集的产生 Procedure apriori_gen (F_{k-1})

② $F_{k-1} \times F_1$：用其他频繁项来扩展每个频繁 (k-1)项集



比蛮力方法有明显改进，但仍有可能产生不必要的候选和重复的候选项集

候选剪枝

项集
{面包，尿布，牛奶}

这种方法将产生 $O(|F_{k-1}| \times |F_1|)$个候选 k 项集

总复杂度为 $O(\Sigma_k k|F_{k-1}||F_1|)$

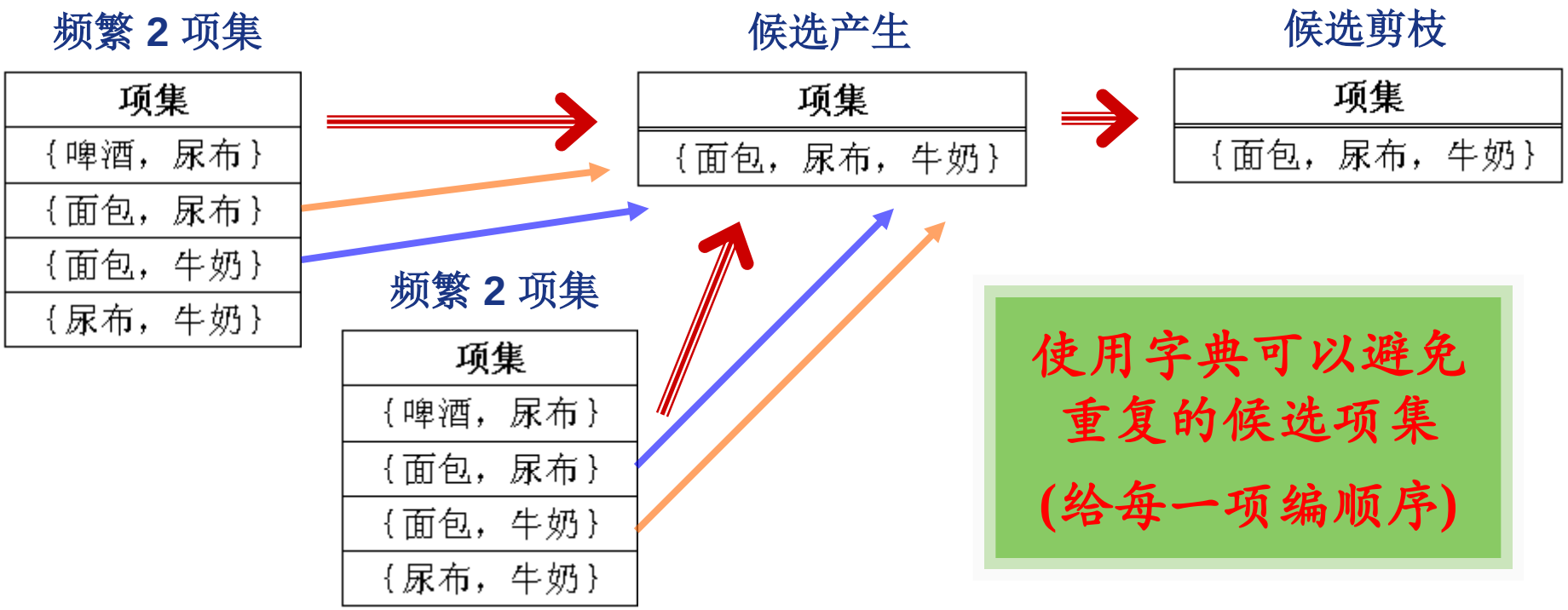
Apriori算法

- 候选项集的产生 Procedure apriori_gen (F_{k-1})

③ $F_{k-1} \times F_{k-1}$：合并一对频繁 (k-1) 项集，，仅当它们的前 (k-1) 个项都相同

令 $A=\{a_1, a_2, \ldots, a_{k-1}\}$ 和 $B=\{b_1, b_2, \ldots, b_{k-1}\}$ 是一对频繁 (k-1)项集，合并 A 和 B，如果它们满足如下条件

$$a_i = b_i \quad (i=1, 2, \ldots, k-2) \quad \text{并且 } a_{k-1} \neq b_{k-1}$$

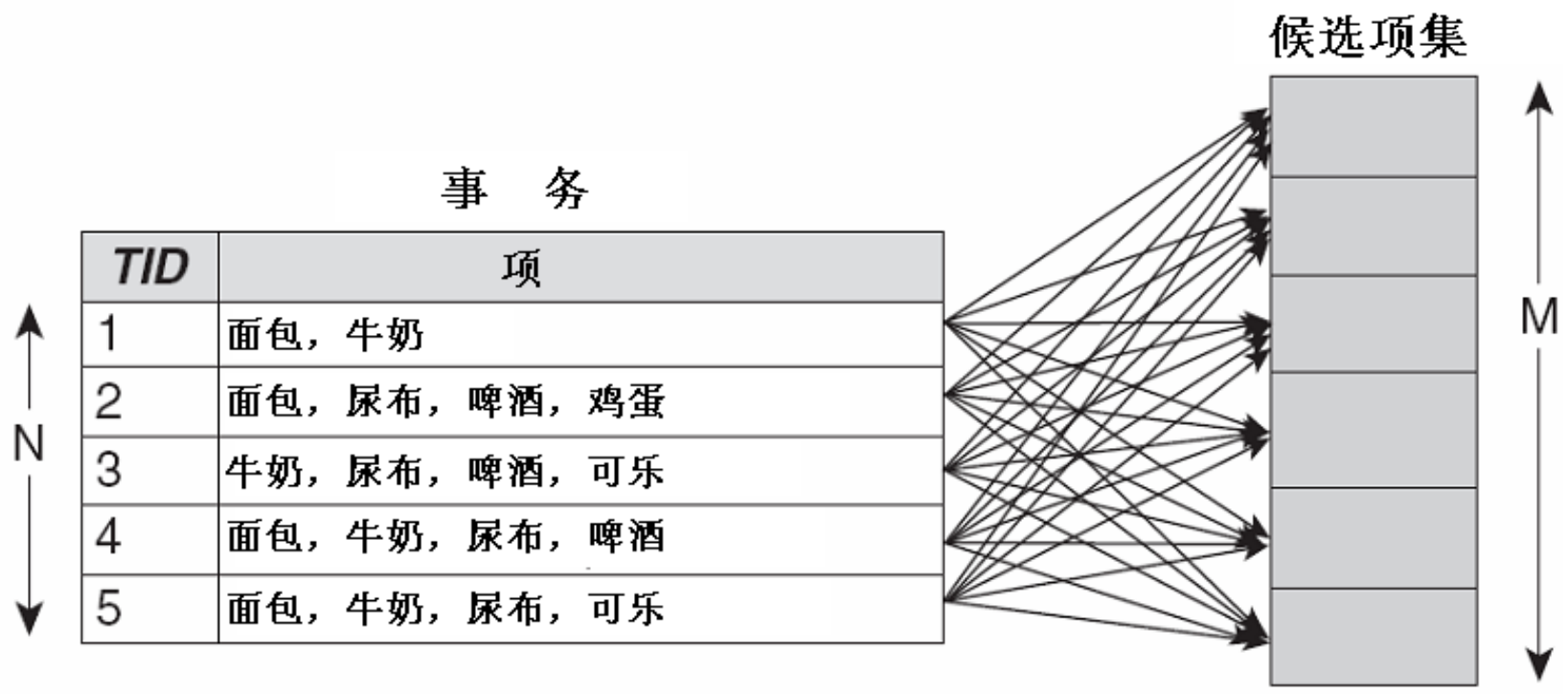


Apriori算法

- 候选项集的剪枝（支持度计数 → 减少比较次数）

➢ 支持度计数过程就是确定在 apriori_gen 函数的候选项剪枝步骤保留下来的每个候选项集出现的频繁程度

① 蛮力方法：将每个事务与所有的候选项集进行比较，并且更新包含在事务中的候选项集的支持度计数。



Apriori算法

- 候选项集的剪枝（支持度计数 → 减少比较次数）

② Hash树方法：枚举每个事务所包含的项集，并且利用它们更新对应的候选项集的支持度。

- 将候选项集划分为不同的桶，并存放在Hash树中（通常需要编码）
- 系统枚举事务所包含的项集
- 在支持度计数期间，包含在事务中的项集也散列到相应的桶中与候选项集进行匹配，如果匹配成功，则相应候选项集的支持度计数增值

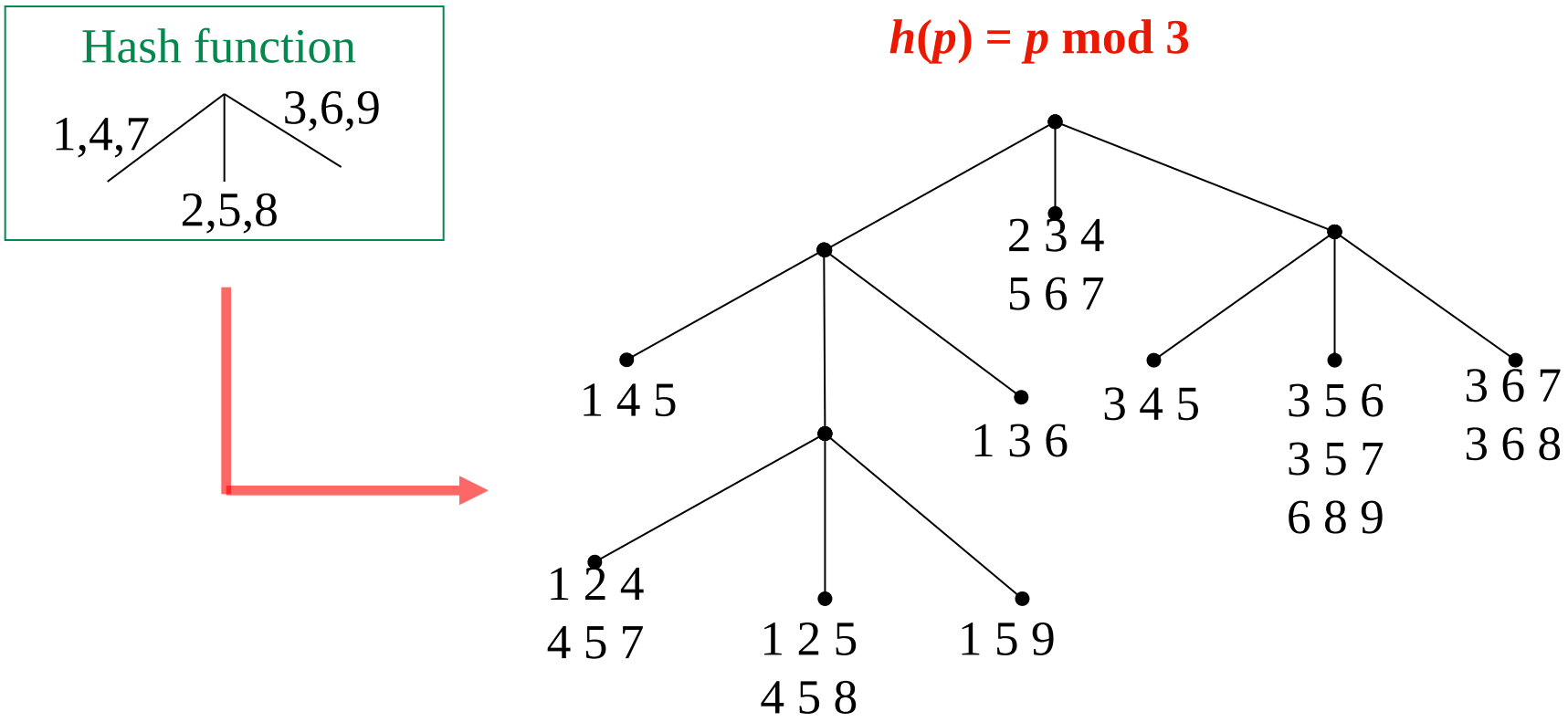
这种方法不是将事务中的每个项集与所有的候选项集进行比较，而是将它与同一桶内候选项集进行匹配

Apriori算法

- 将候选项集划分为不同的桶，并存放在Hash树中（通常需要编码）

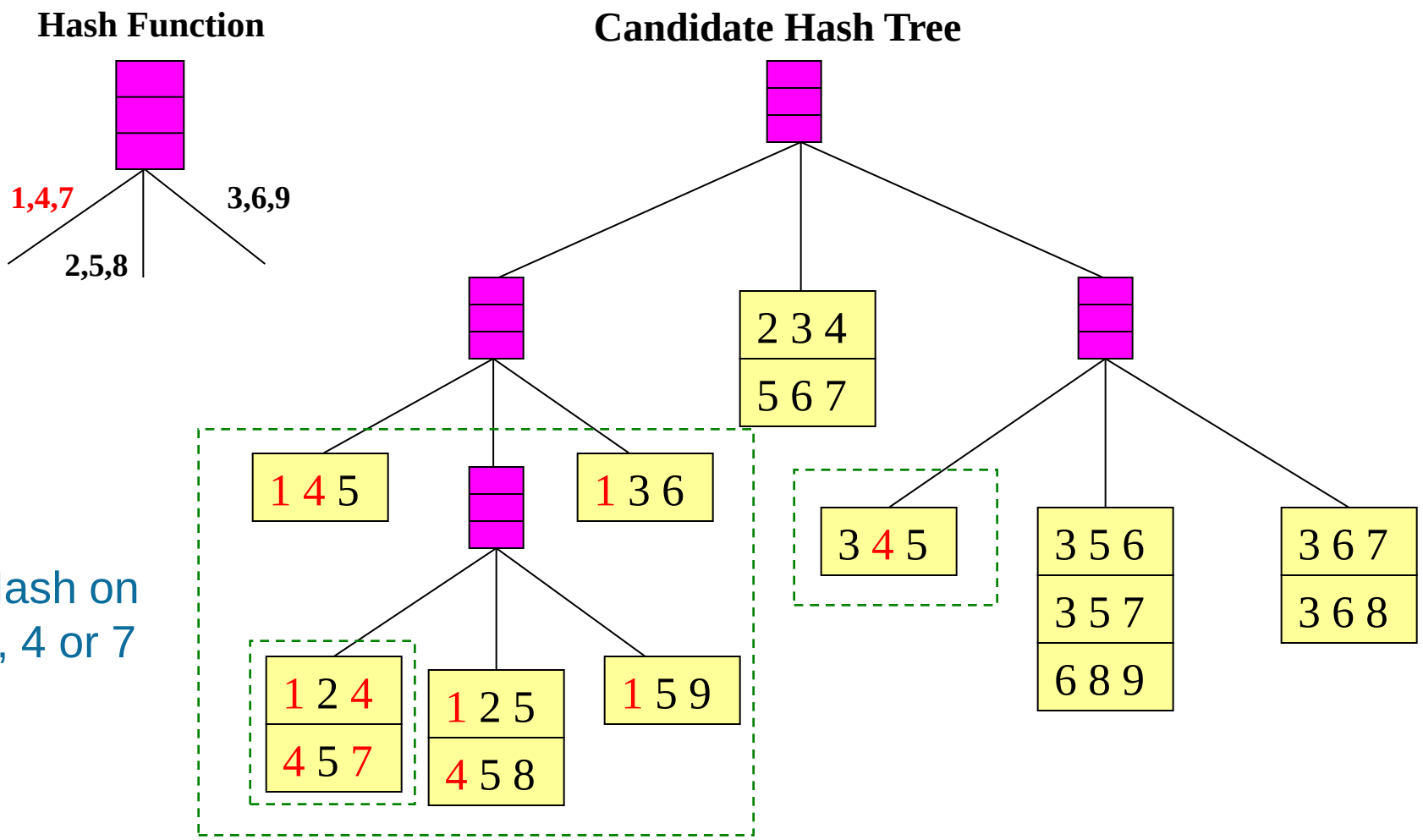
■ 示例　假设以下为 15个三项候选项集

{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}



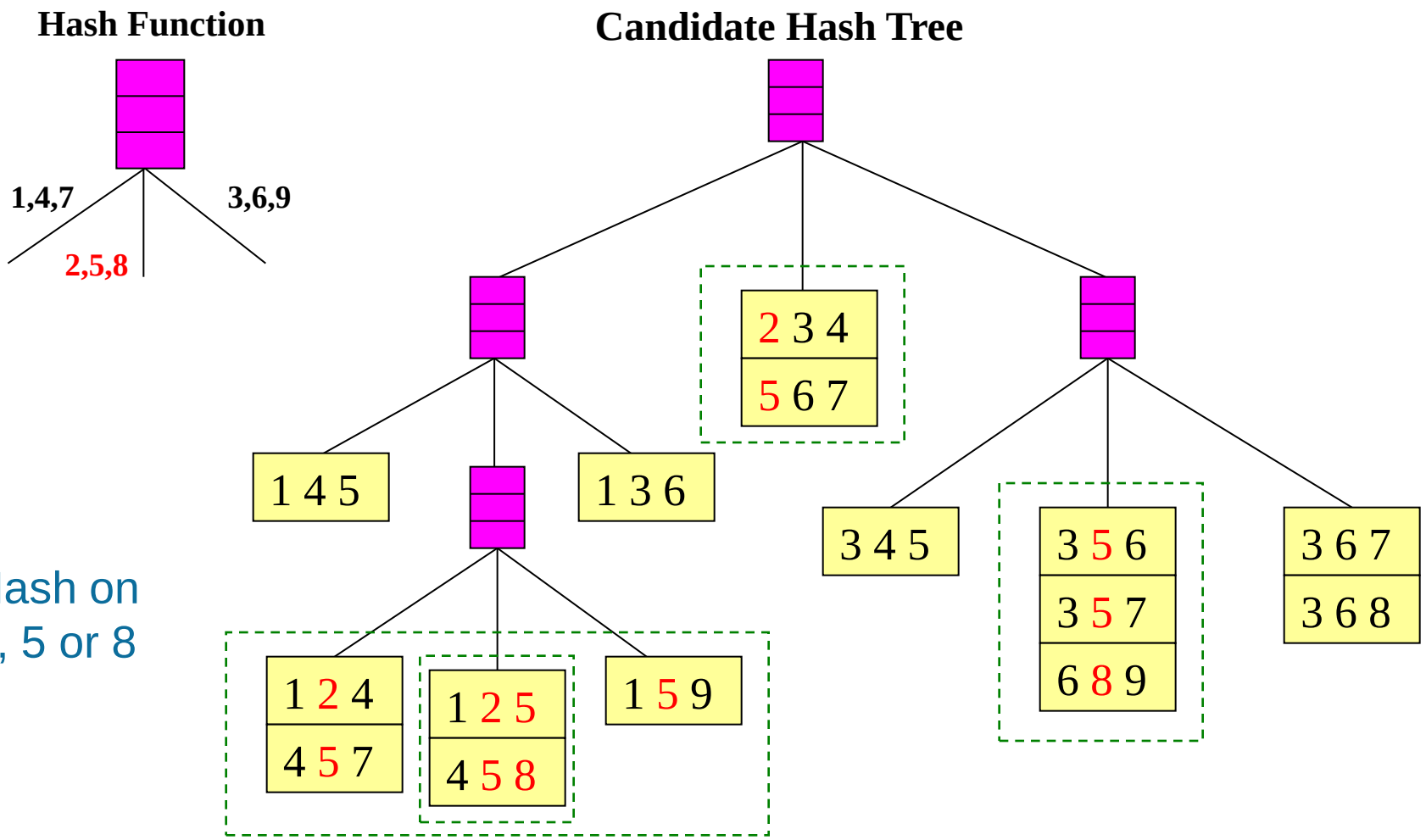
Apriori算法

{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}



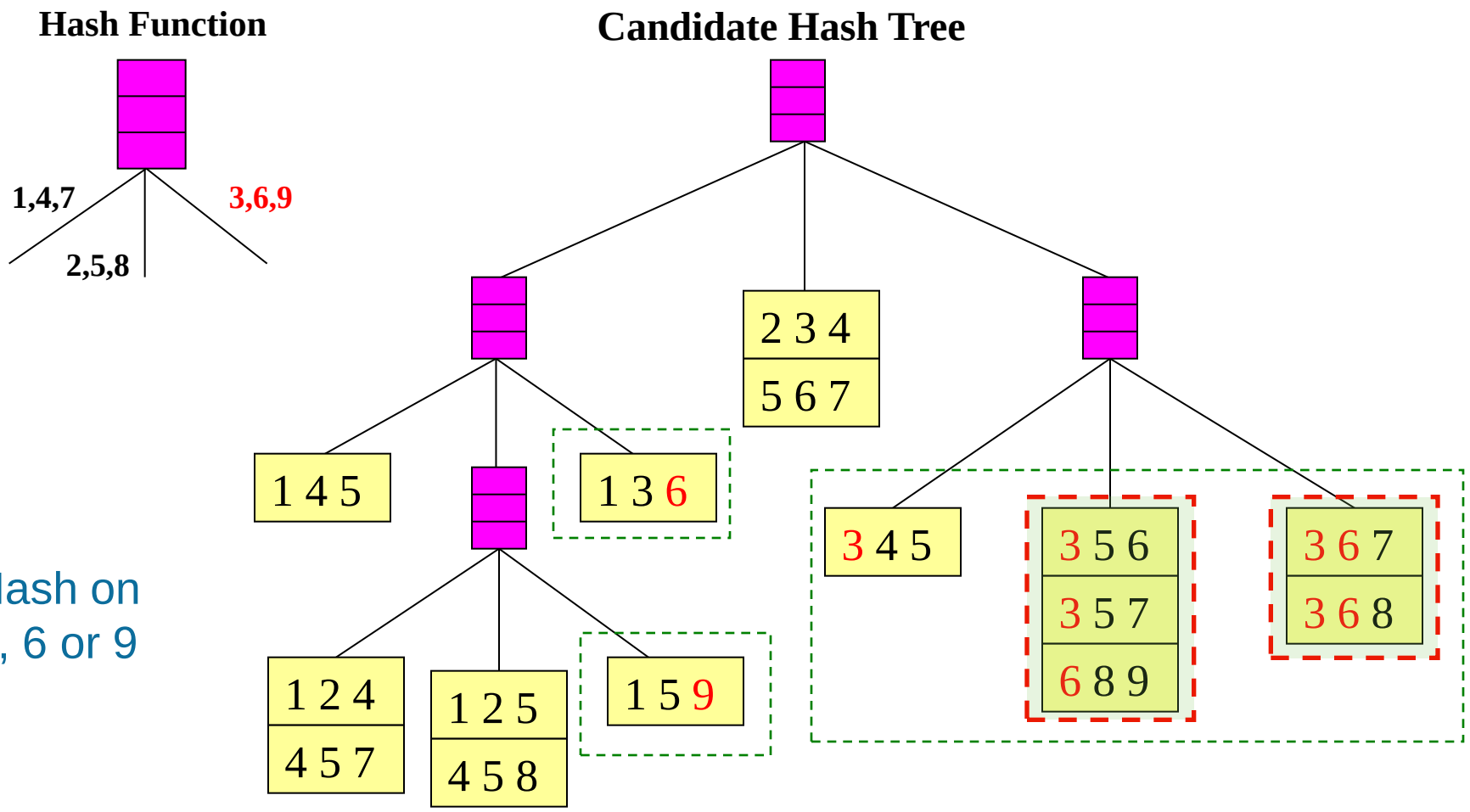
Apriori算法

{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}



Apriori算法

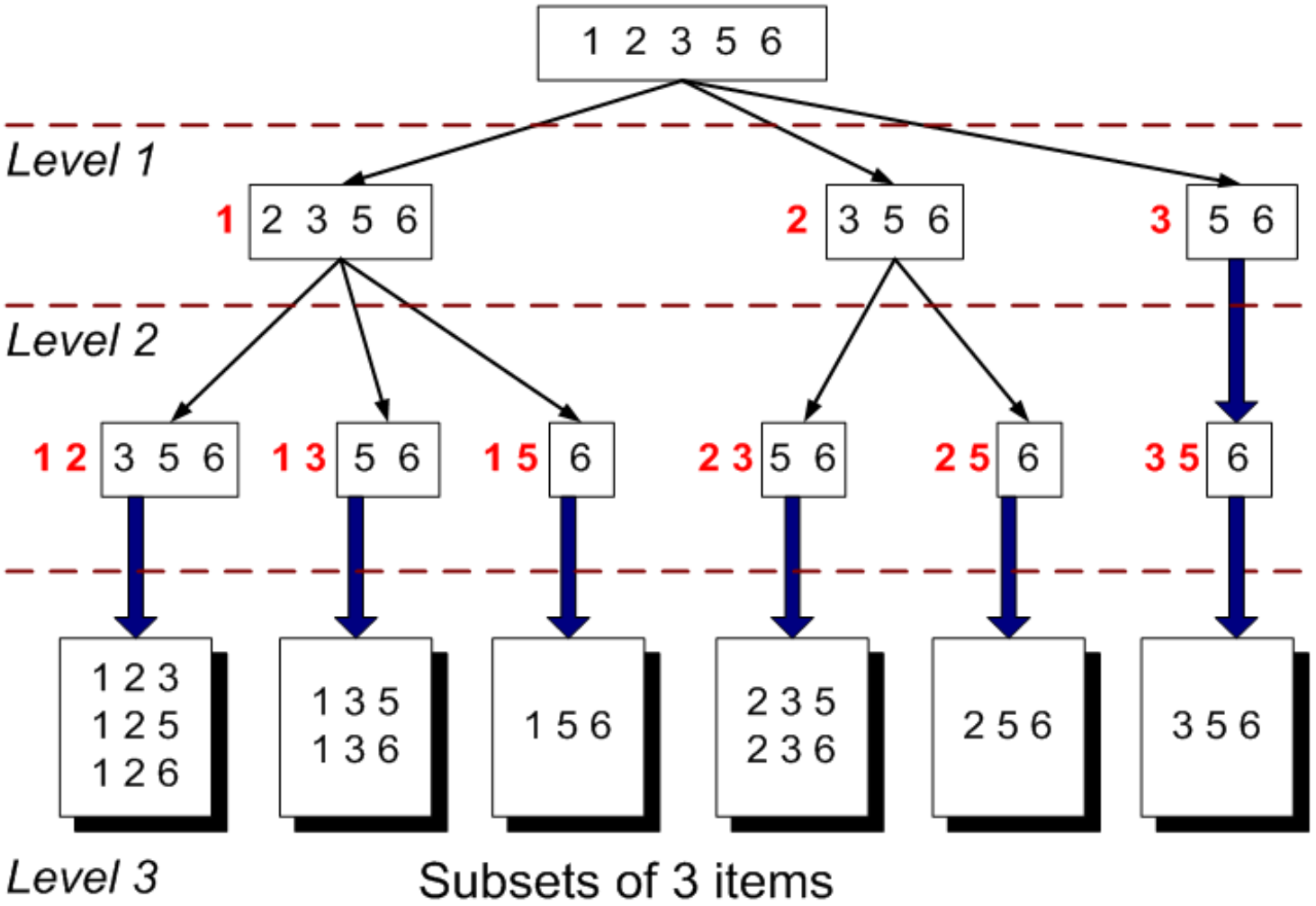
{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}



Apriori算法

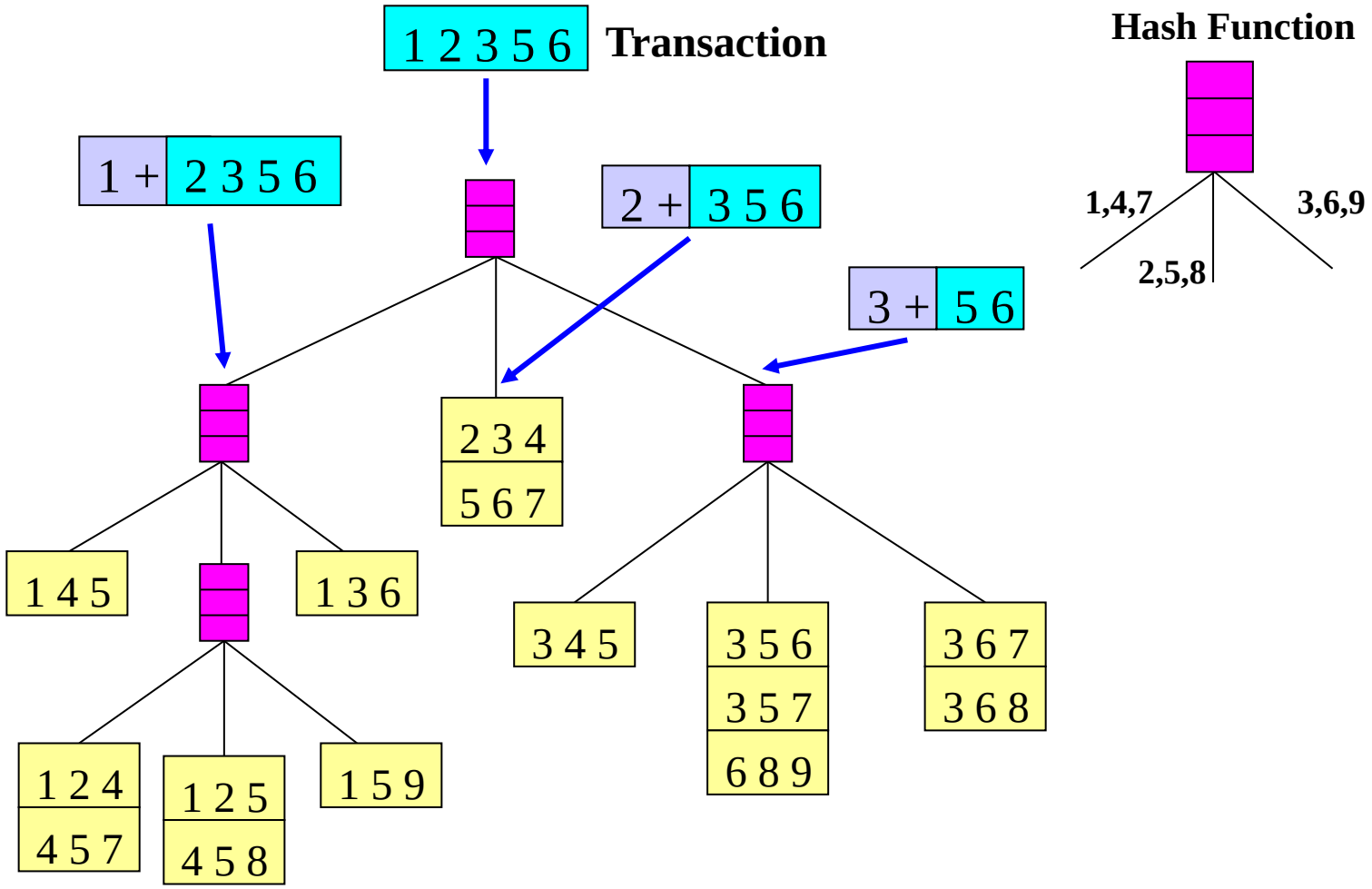
- 系统枚举事务所包含的项集

- **示例** 假设事务 t = { 1, 2, 3, 5, 6}，系统枚举该事务所包含的三项集

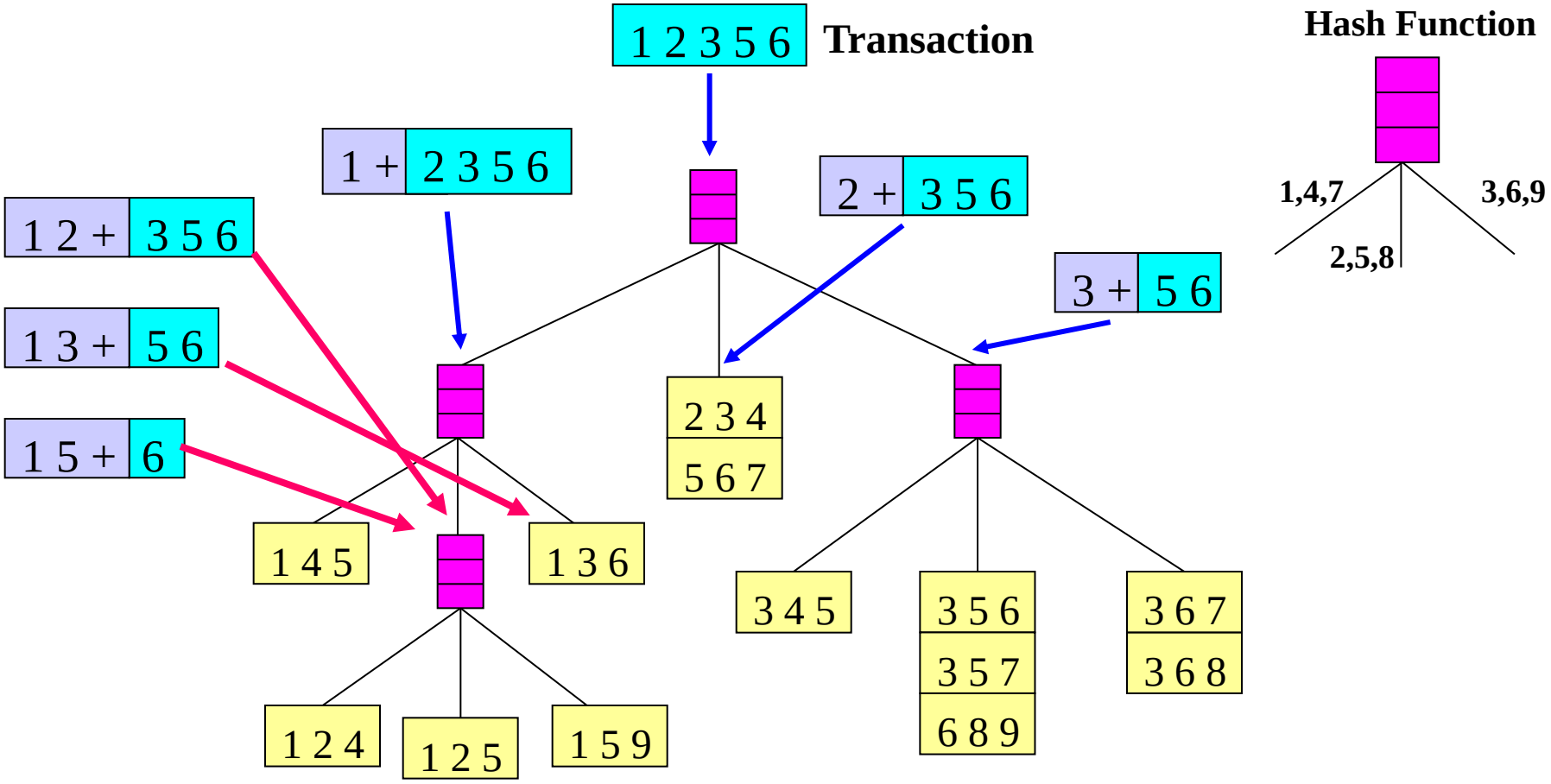


Apriori算法

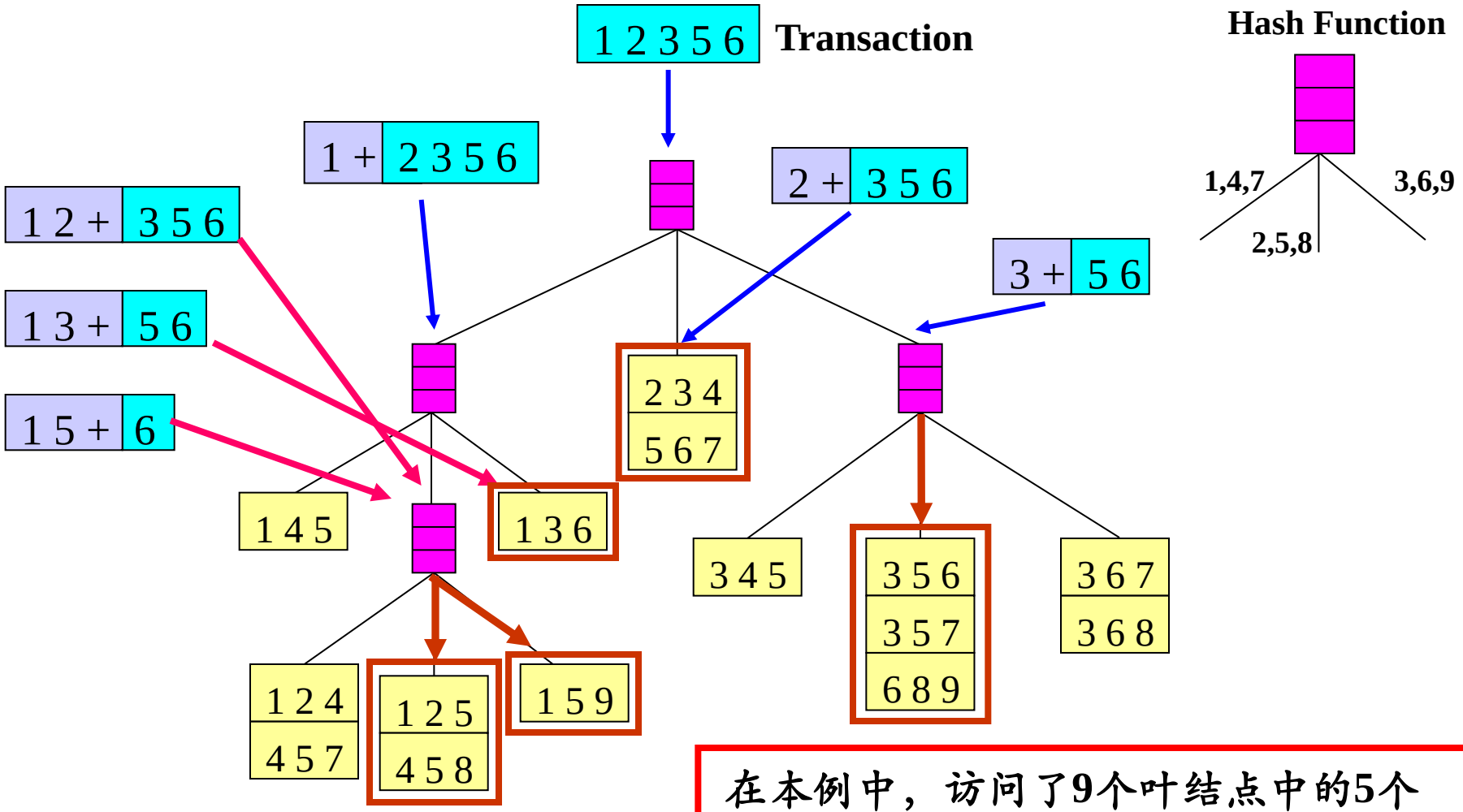
- 在支持度计数期间，包含在事务中的项集也散列到相应的桶中与候选项集进行匹配，如果匹配成功，则相应候选项集的支持度计数增值



Apriori算法


1 2 3 5 6
Transaction
Hash Function
1,4,7
2,5,8
3,6,9
1 +
2 3 5 6
2 +
3 5 6
3 +
5 6
1 2 +
3 5 6
1 3 +
5 6
1 5 +
6
2 3 4
5 6 7
1 4 5
1 3 6
3 4 5
3 5 6
3 5 7
6 8 9
3 6 7
3 6 8
1 2 4
1 2 5
1 5 9

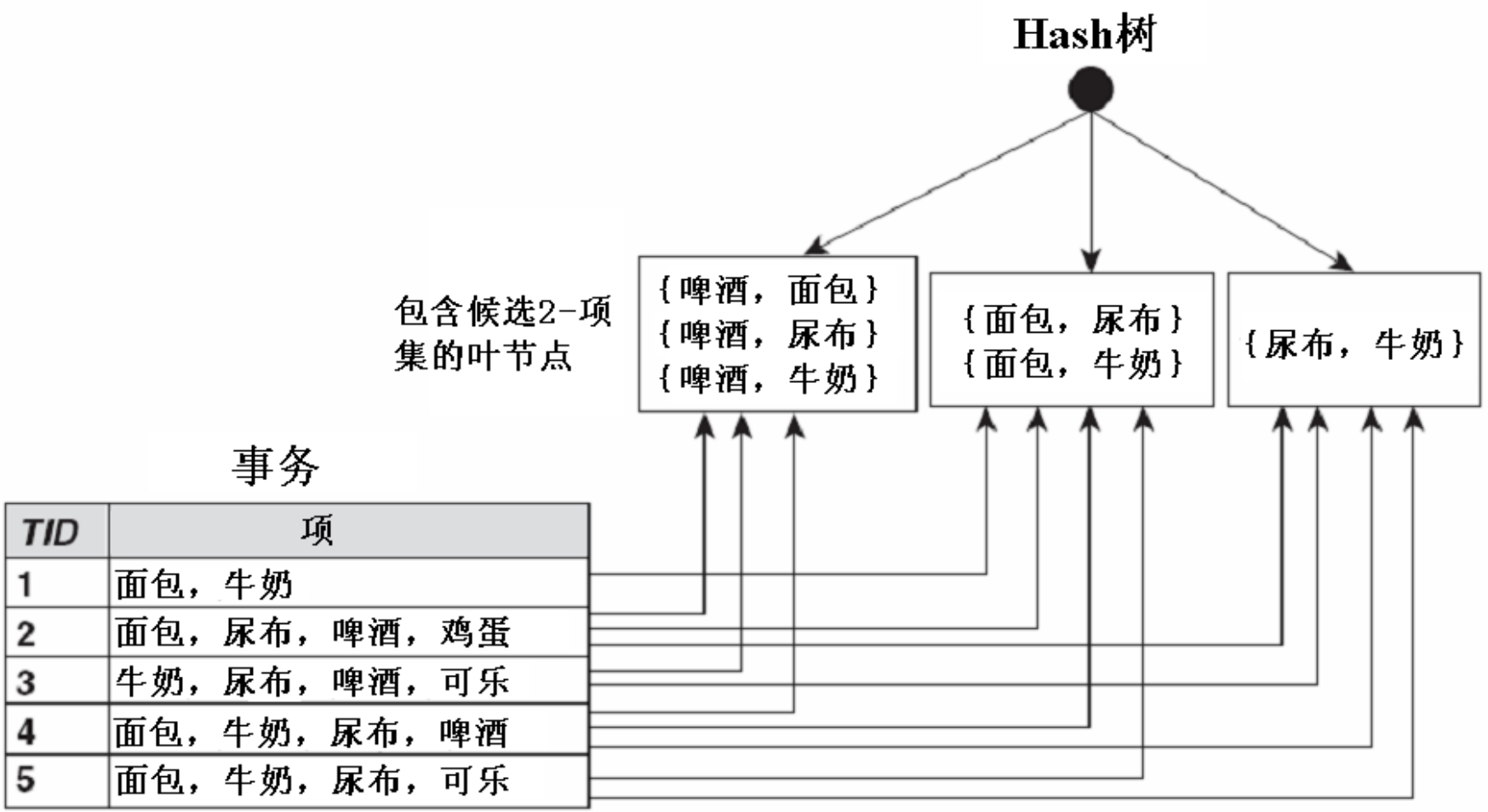
Apriori算法



在本例中，访问了9个叶结点中的5个

15个候选项集中的9个与该事务进行比较

Apriori算法



事务

TID	项
1	面包，牛奶
2	面包，尿布，啤酒，鸡蛋
3	牛奶，尿布，啤酒，可乐
4	面包，牛奶，尿布，啤酒
5	面包，牛奶，尿布，可乐

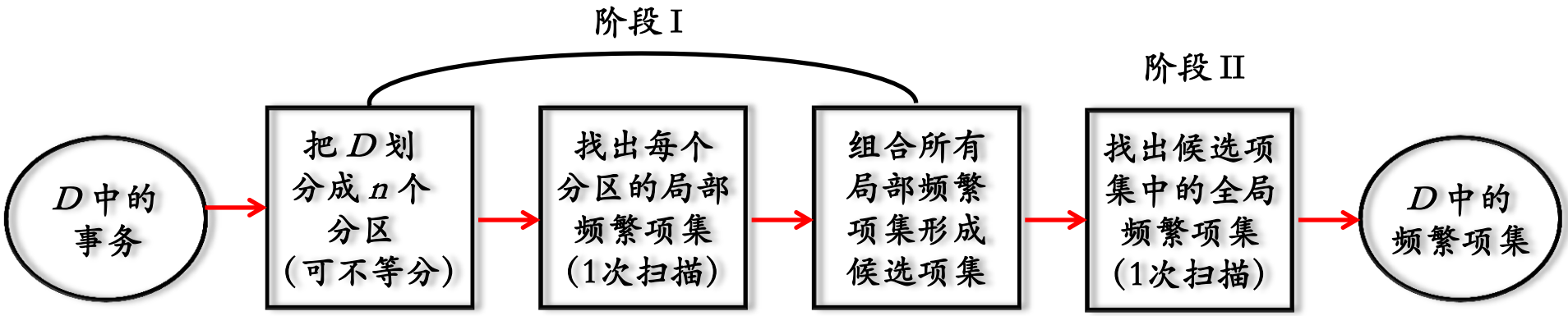
Apriori算法

- **提高 Apriori算法的效率**

- 样本抽样：在给定数据的一个子集上挖掘，牺牲精度换取效率
- 事务压缩：不包含任何频繁 k 项集的事务不可能包含任何频繁 $(k+1)$ 项集。因此，在产生 j 项集 $(j > k)$ 的数据库扫描时不再需要扫描它们
- 划分：为寻找候选项集划分数据

  如果 D 中事务的最小支持度为 min_sup, 则每个分区的最小支持度计数为

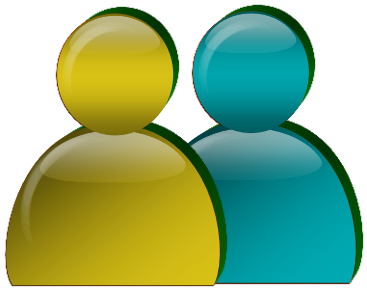
  $min\_sup \times$ 该分区中的事务数

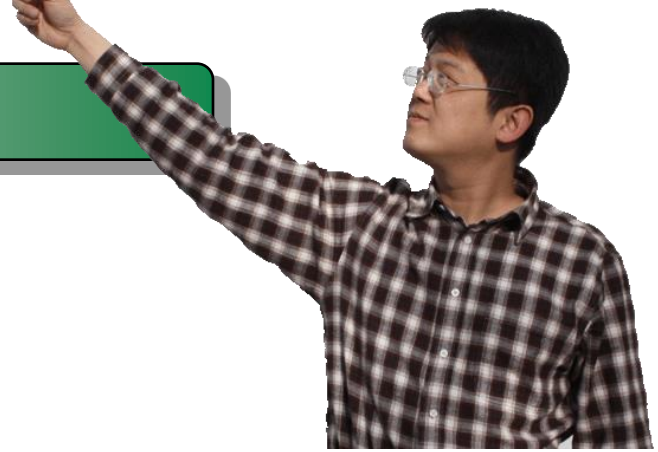


局部频繁项集不一定是 D 的频繁项集，但 D 的任何频繁项集必须作为局部频繁项集至少出现在一个分区中

关联规则模式发现与识别

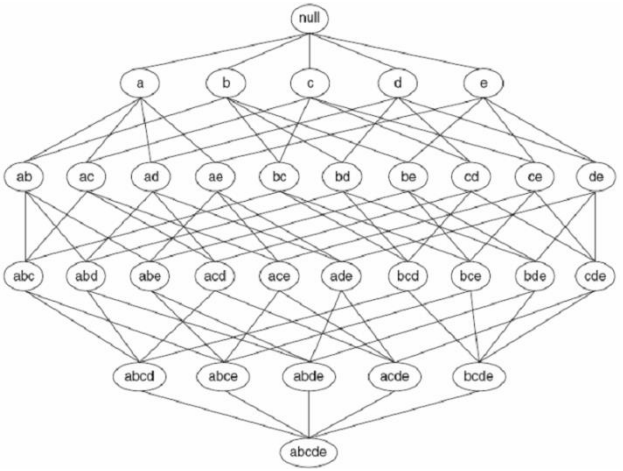
1. 基本概念
2. Apriori算法
3. 发现频繁项集的其他方法
4. 关联规则的扩展





发现频繁项集的其他方法

- **项集格遍历**

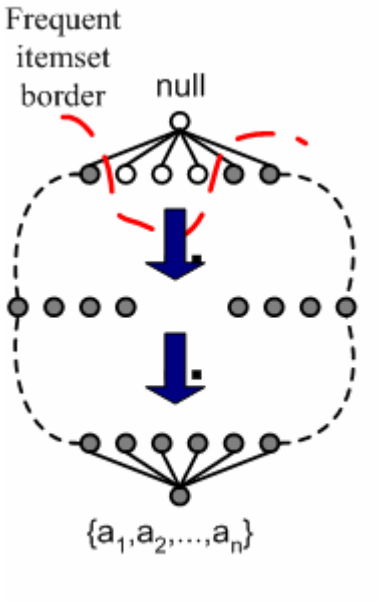


➢ 频繁项集的搜索可以看作遍历右图中的项集格

➢ 算法使用的搜索策略指明了频繁项集产生过程中如何遍历格结构

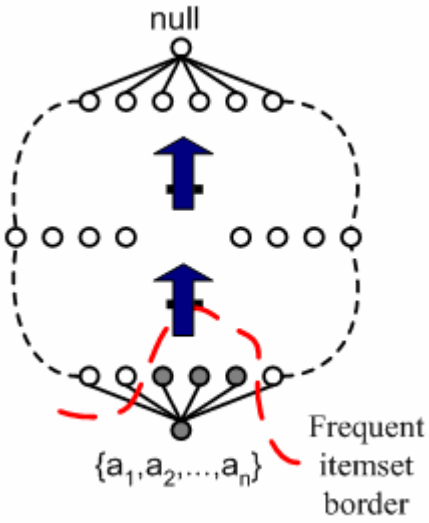
① 一般到特殊与特殊到一般

频繁项集的长度不是太长



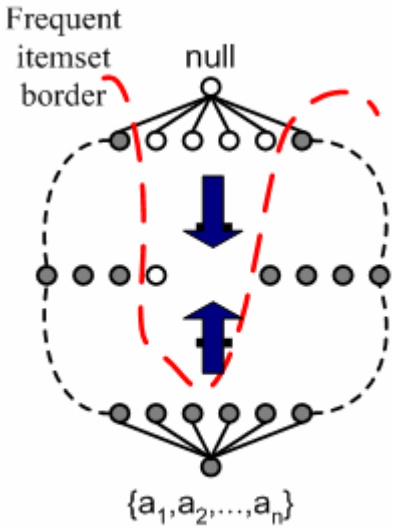
(a) General-to-specific

有利于发现稠密事务中的最大频繁项集



(b) Specific-to-general

更快搜索速度更多存储空间

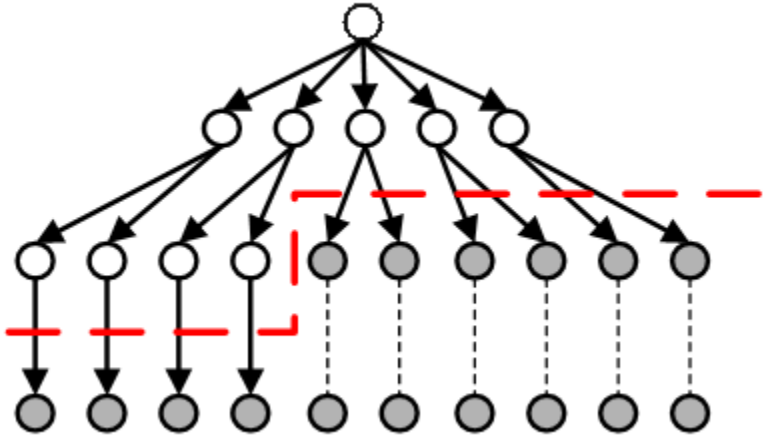


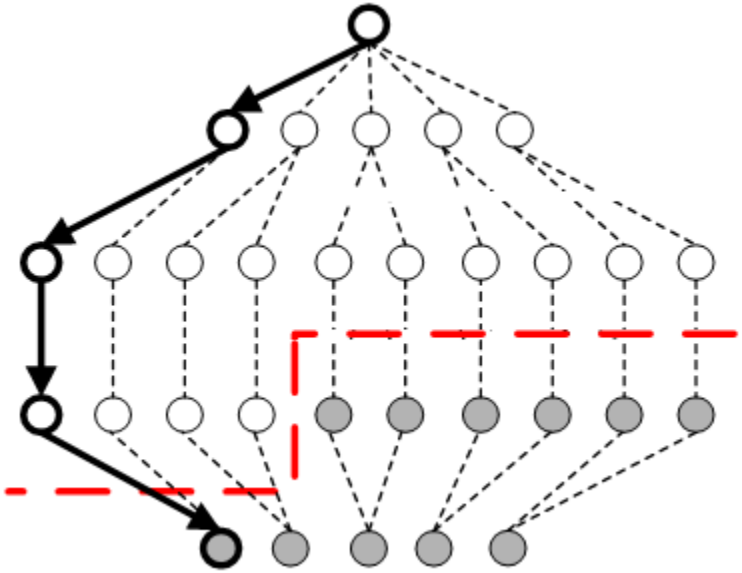
(c) Bidirectional

发现频繁项集的其他方法

- 项集格遍历

② 宽度优先与深度优先



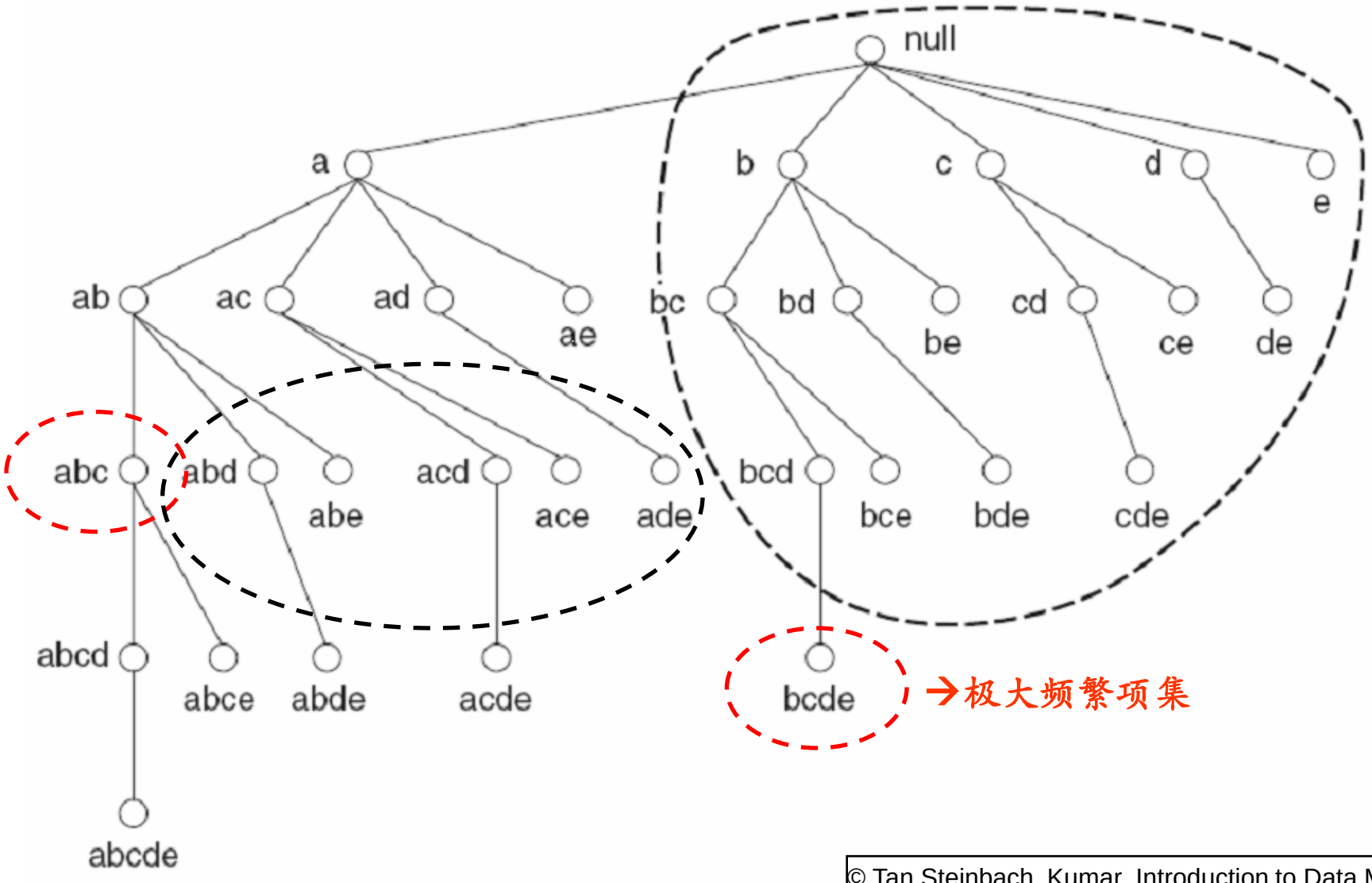


✓ 深度优先比宽度优先更快地检测到频繁项集边界

✓ 一旦发现一个最大频繁项集，就可以在它的子集上进行剪枝

发现频繁项集的其他方法

- 示例

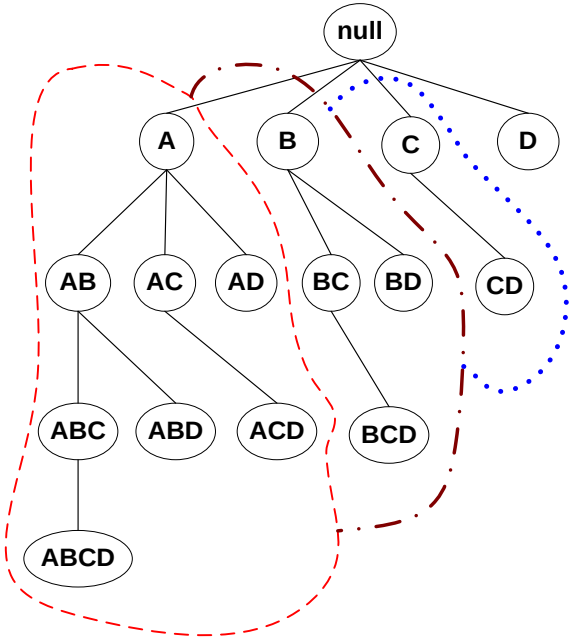


发现频繁项集的其他方法

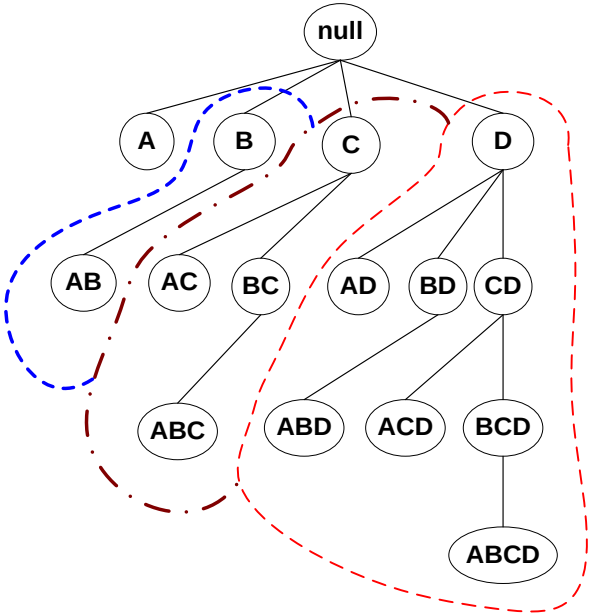
- 项集格遍历

③ 等价类

- 将格划分为多个不相交的结点组（或等价类）
- 依次在每个等价类内搜索频繁项集



(a) Prefix tree



(b) Suffix tree

发现频繁项集的其他方法

- 使用垂直数据格式挖掘频繁项集

TID集	项
1	a, b, c
2	b, c, d
3	b, d
4	a, c, d

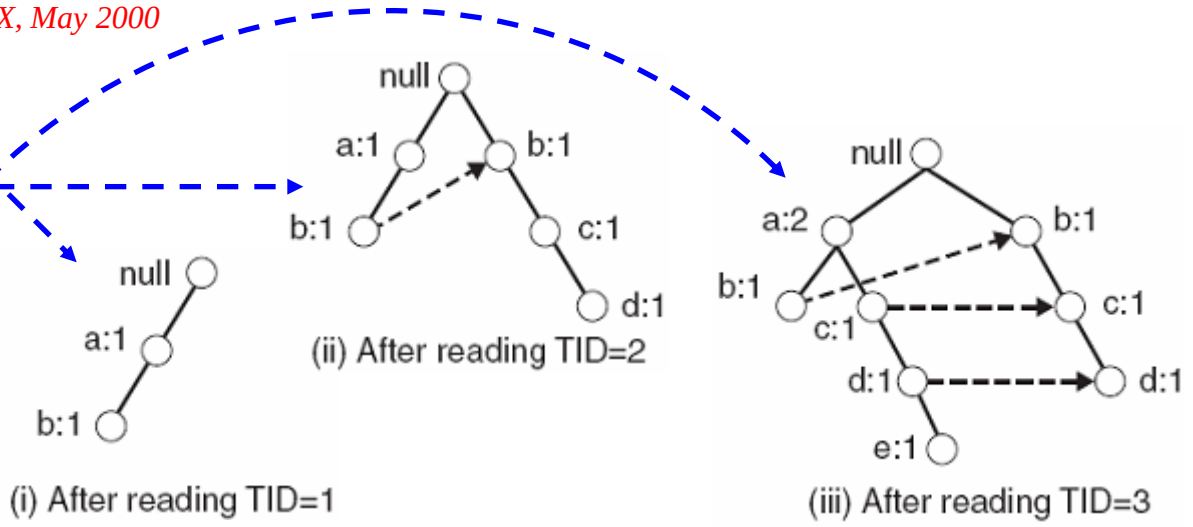
项集	TID集
a	1, 4
b	1, 2, 3
c	1, 2, 4
d	2, 3, 4

- FP增长算法（frequent-pattern growth；直接产生频繁项集）

J.Han, J.Pei, and Y.Yin. Mining Frequent Patterns without Candidate Generation. In Proc. ACM-SIGMOD Int. Conf. on Management of Data (SIGMOD'00), pages 1-12, Dallas, TX, May 2000

① 构造FP树

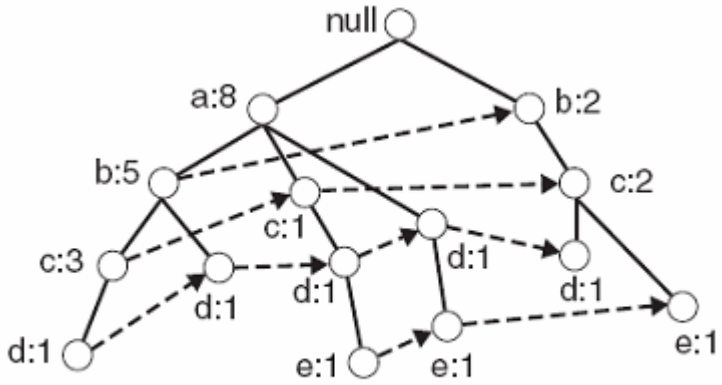
TID	Items
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}



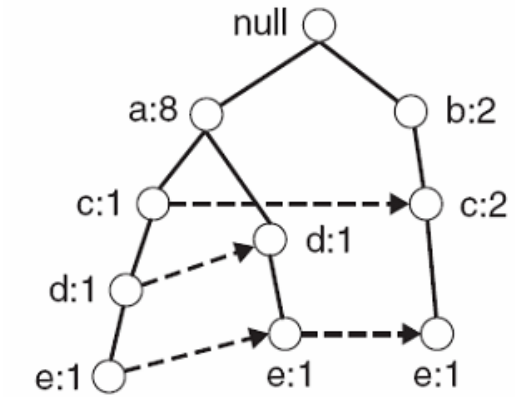
发现频繁项集的其他方法

TID	Items
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}

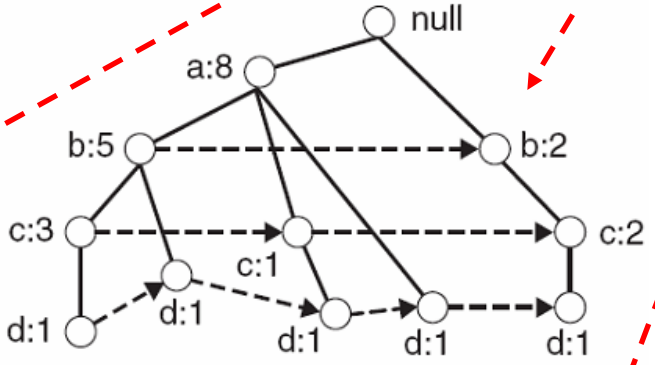
② 产生频繁项集



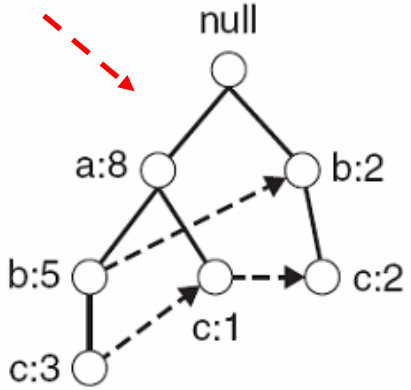
(iv) After reading TID=10



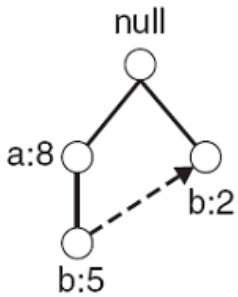
(a) Paths containing node e



(b) Paths containing node d



(c) Paths containing node c



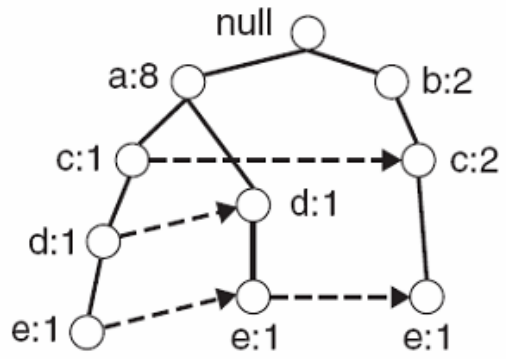
(d) Paths containing node b

(e) Paths containing node a

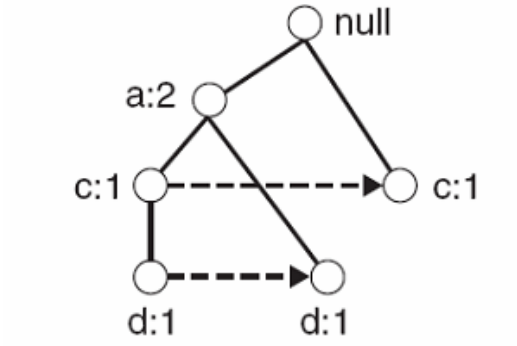
发现频繁项集的其他方法

- 示例

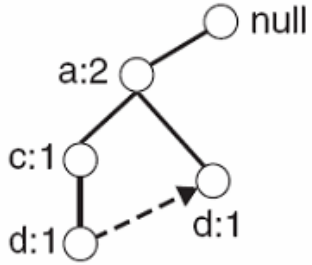
假定最小支持度为 2



(a) Prefix paths ending in e

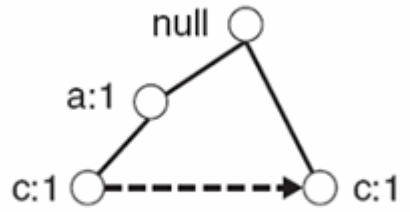


(b) Conditional FP-tree for e



(c) Prefix paths ending in de

(d) Conditional FP-tree for de



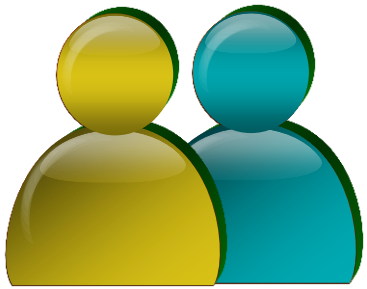
(e) Prefix paths ending in ce

(f) Prefix paths ending in ae

后缀	频繁项集
e	

关联规则模式发现与识别

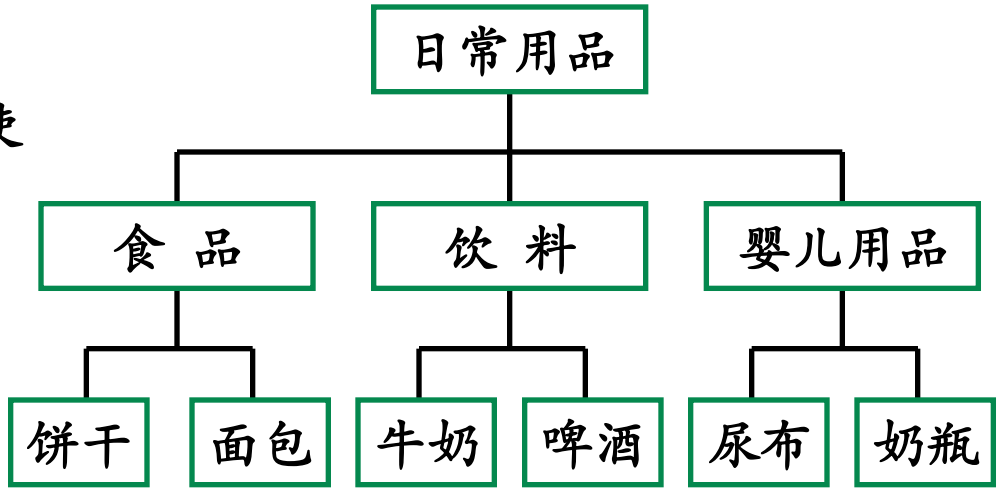
1. 基本概念
2. Apriori算法
3. 发现频繁项集的其他方法
4. 关联规则的扩展



关联规则的扩展

- **多层次关联规则**

  - 将数据项沿概念层次向上提升，使多个数据项合并为一个新的概念，从而使其支持度增加



面包 => 牛奶

面包 => 饮料

一致支持度 VS. 递减支持度

- **多维关联规则**

年龄（X,“20 ～30”）∩ 职业（X,“学生”）=> 购买（X,“笔记本电脑”）

多维关联规则挖掘是发现频繁谓词集

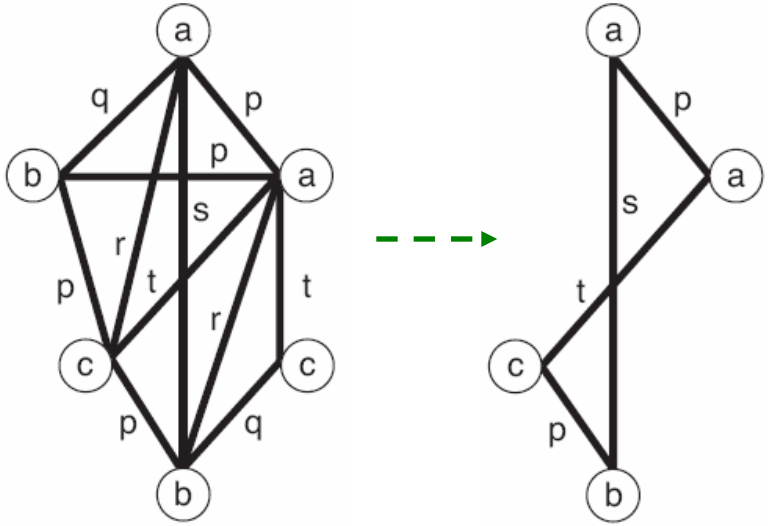
- **定量关联规则**

  - 从包含连续属性的数据集中挖掘关联规则，但可以通过将其离散化而转化为布尔关联规则

静态离散化 VS. 动态离散化

关联规则的扩展

- **加权关联规则**
  - 为了反映各个项目的重要性不同，可以给每个项目分配一个反映其重要程度的权值，并给出项集和规则的加权支持度，从而扩展现有的关联规则问题模型
- **频繁子图挖掘**
  - 给定图的集合 *S* 和用户指定的最小支持度阈值 *minsup*，频繁子图挖掘的目标是在 *S* 中发现一组公共子结构（子图），满足支持度大于或等于 *minsup*



- **序列模式发现**
  - 给定序列数据集 *D* 和用户指定的最小支持度阈值 *minsup*，序列模式发现的任务是找出支持度大于或等于 *minsup* 的所有序列

关联规则的扩展

- **非频繁模式挖掘**

- 非频繁模式是一个项集或规则，其支持度小于阈值 *minsup*

- 有助于识别竞争项（competing item）

  DVD ←→ VCR

  茶 ←→ 咖啡

  黄油 ←→ 人造黄油

  普通苏打水←→ 节食苏打水

  台式计算机 ←→ 便携式计算机

- 有助于发现罕见事件或例外情况

  {学习刻苦 = yes} ←→ {学习刻苦= yes，成绩很差 = yes}

关联规则的扩展

- 哪些模式是有趣的：模式评估方法

➢ 支持度－置信度框架的问题（强关联规则也可能是无趣的甚至是误导的）

事物总数	计算机游戏	录像	同时包含计算机游戏和录像
10000	6000	7500	4000

minsup = 30%

minconf = 60%

购买（*X*，“计算机游戏”）=> 购买（*X*，“录像”）

[support = 40% , confidence = 67%]

购买计算机游戏的顾客同时也会偏向于购买录像（强规则）

购买录像的概率 75% > 67%

玩计算机游戏 *PK* 看录像

（负相关）

关联规则的扩展

- 哪些模式是有趣的：模式评估方法

  - 从关联分析到相关分析

提升度（lift）

χ^2 度量

A => B [support, confidence, correlation]

相关系数

协方差

$P(AB) = P(A)P(B)$：项集 A 的出现独立于项集 B 的出现

$P(AB) \neq P(A)P(B)$：项集 A 和 B 是依赖 (dependent) 和相关的 (correlated)

提升度（lift） $lift(A,B) = \dfrac{P(AB)}{P(A)P(B)} = \dfrac{P(A|B)}{P(A)}$

- >1 正相关
- <1 负相关

事物总数	10000
计算机游戏	6000
录像	7500
同时包含计算机游戏和录像	4000

	game	$\overline{\text{game}}$	$\sum_{row}$
video			
$\overline{\text{video}}$			
$\sum_{col}$			

关联规则的扩展

- 哪些模式是有趣的：模式评估方法

➢ 从关联分析到相关分析

事物总数	10000
计算机游戏	6000
录像	7500
同时包含计算机游戏和录像	4000

	game	$\overline{\text{game}}$	$\sum_{row}$
video	4000	3500	7500
$\overline{\text{video}}$	2000	500	2500
$\sum_{col}$	6000	4000	10000

提升度（lift） $lift(A,B) = \dfrac{P(AB)}{P(A)P(B)} = \dfrac{P(A|B)}{P(A)}$

> 1 正相关

< 1 负相关

$P(\{game\}) = 0.60$

$P(\{video\}) = 0.75$

$P(\{game, video\}) = 0.40$

$$lift(game, video) = \frac{0.40}{0.60 \times 0.75}$$

$$= 0.89 < 1$$

关联规则的扩展

- 哪些模式是有趣的：模式评估方法

  - 从关联分析到相关分析

$$\chi^2 = \sum \frac{(\text{观测值} - \text{期望值})^2}{\text{期望值}}$$

	game	$\overline{\text{game}}$	$\sum_{row}$
video	4000	3500	7500
$\overline{\text{video}}$	2000	500	2500
$\sum_{col}$	6000	4000	10000

$$\text{期望值 } \mathbf{E}_{ij} = \frac{count(A=ai) \times count(B=b_j)}{n}$$

相关分析如此管用，为什么还需要关联分析？

$$\mathbf{E}_{game,\ video} = \frac{count(game) \times count(video)}{n} = \frac{6000 \times 7500}{10000} = 4500$$

	game	$\overline{\text{game}}$	$\sum_{row}$
video	4000 (4500)	3500 (3000)	7500
$\overline{\text{video}}$	2000 (1500)	500 (1000)	2500
$\sum_{col}$	6000	4000	10000

$\chi^2 = 55.6 > 10.828$

自由度 (2-1) × (2-1) = 1

在0.001置信水平下

拒绝独立假设的值

4000 < 4500

关联规则的扩展

- 哪些模式是有趣的：模式评估方法

零事物
零不变性（度量）
不平衡比（IR）

- 支持度－置信度框架的扩展
  - 模式评估度量比较

- 全置信度（all_confidence，“A⇒B”和“B⇒A”的最小置信度）

$$all\_conf(A,B) = \frac{sup(AB)}{max\{sup(A), sup(B)\}} = min\{P(A|B), P(B|A)\}$$

- 最大置信度（max_confidence，“A⇒B”和“B⇒A”的最大置信度）

$$max\_conf(A,B) = \frac{sup(AB)}{min\{sup(A), sup(B)\}} = max\{P(A|B), P(B|A)\}$$

- Kulczynski度量（Kulc）

$$Kulc(A,B) = \frac{1}{2}(P(A|B) + P(B|A))$$

- 余弦度量（调和提升度）

$$cosine(A,B) = \frac{sup(AB)}{\sqrt{sup(A) \times sup(B)}} = \frac{P(AB)}{\sqrt{P(A) \times P(B)}} = \sqrt{P(A|B) \times P(B|A)}$$

- 度量值仅受A、B和AB的支持度的影响，不受事物总个数的影响
- 每个度量值都遍取 0 ~ 1

关联规则的扩展

畏惧错误就是毁灭进步。

—— Robert Whitehead

英国工程师（现代鱼雷之父）（1823 – 1905）

在飞速发展的科学上，随着年龄的增长，我们的使命是：看到纠正旧有观念的新发现时，我们应感到愉快，并在教学的过程中以自己的学生为师。这是预防中年时期老顽固病症的唯一有效措施。

—— Hans Zinsser

美国微生物学家和免疫学家（1878 – 1940）

Pattern Recognition & Machine Learning

To be continued ……

wangyong@ucas.ac.cn

http://people.ucas.ac.cn/~wangyong