数 据 挖 掘

Data Mining

主讲: 张仲楠 教授

厦門大學
XIAMEN UNIVERSITY

关联分析

目 录

1.基本概念

1. 总体思路

- 关联分析(association analysis)：用于发现隐藏在大型数据集中有意义的联系。
- 频繁项集(frequent itemset)：用出现在许多事务中的项集来表示所发现的联系。
- 关联规则(association rule)：表示两个项集之间的关系。

事务

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

{Diaper} → {Beer}
{Milk, Bread} → {Eggs,Coke}
{Beer, Bread} → {Milk}

1.基本概念

1. 总体思路

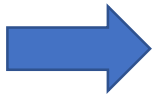
- 在进行关联分析时，需要处理两个关键的问题:
- 第一，从大型事务数据集中发现模式可能在计算上要付出很高的代价;
- 第二，所发现的某些模式可能是虚假的(仅仅是偶然发生的)，甚至对于非虚假的模式来说，它们的意义也会有所不同的。
- 思路:
  - （1）找到有效地挖掘这种模式的算法
  - （2）如何评估所发现的模式，以避免产生虚假结果，并根据一些兴趣度量来对模式进行排序

1.基本概念

2. 基本定义

■ 二元表示：购物篮数据可以用下表所示的二元形式来表示，其中每行对应一个事务，而每列对应一个项。项可以用二元变量表示，如果项在事务中出现，则它的值为1,否则为0。因为通常认为项在事务中出现比不出现更重要，因此项是非对称(asymmetric)二元变量。

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke



TID	Beer	Bread	Coke	Diaper	Egg	Milk
1	0	1	0	0	0	1
2	1	1	0	1	1	0
3	1	0	1	1	0	1
4	1	1	0	1	0	1
5	0	1	1	1	0	1

1.基本概念

2. 基本定义

- 令$I = \{i_1, i_2, \cdots, i_d\}$是购物篮数据中所有项的集合，而$T = \{t_1, t_2, \cdots, t_N\}$是所有事务的集合。
- 每个事务t_i包含的项集都是I的子集。
- 在关联分析中，包含零个或多个项的集合称为项集(itemset)。
- 如果一个项集包含k个项，则称它为k-项集。
  - 例如，{啤酒，尿布，牛奶}是一个 3-项集。
- 空集是指不包含任何项的项集。

1.基本概念

TID	Beer	Bread	Coke	Diaper	Egg	Milk
1	0	1	0	0	0	1
2	1	1	0	1	1	0
3	1	0	1	1	0	1
4	1	1	0	1	0	1
5	0	1	1	1	0	1

2. 基本定义

- 如果项集X是事务t_j的子集，则称事务t_j包括项集X。
  - 例如，事务t_2包括项集{面包，尿布}，但不包括项集{面包，牛奶}。
- 项集的一个重要性质是它的支持度(support)计数，即包含某个特定项集的事务个数。项集X的支持度计数$\sigma(X)$可以表示为:

$$\sigma(X) = |\{t_i | X \subseteq t_i, t_i \in T\}|$$

  - 例如：项集{啤酒，尿布，牛奶}的支持度计数为2，因为只有2个事务（t_3和t_4）同时包含这3个项。

1.基本概念

2. 基本定义

- 支持度（support）：某一项集出现的比率

$$s(X) = \frac{\sigma(X)}{N}$$

- 如果$s(X)$比用户定义的阈值 $minsup$ 大，则称项集X是频繁的。
- 关联规则是形如$X \rightarrow Y$的蕴涵表达式，其中X和Y不相交，即$X \cap Y = \emptyset$。
- 关联规则的强度可以用它的支持度和置信度度量。
- 支持度确定规则可以用于给定数据集的频繁程度，而置信度确定Y在包含X的事务中出现的频繁程度。

1.基本概念

TID	Beer	Bread	Coke	Diaper	Egg	Milk
1	0	1	0	0	0	1
2	1	1	0	1	1	0
3	1	0	1	1	0	1
4	1	1	0	1	0	1
5	0	1	1	1	0	1

{牛奶，尿布，啤酒}

{牛奶，尿布}

2. 基本定义

■ 关联规则的支持度(s)和置信度(c)这两种度量的形式定义如下:

$$s(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{N} \qquad c(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}$$

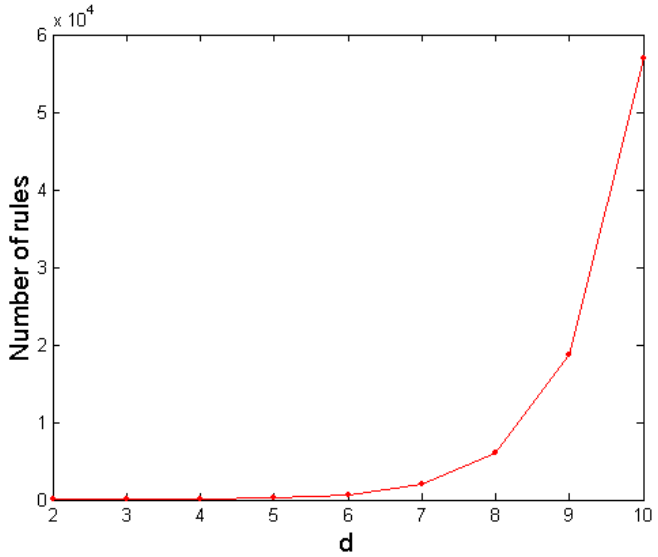
■ 考虑规则{牛奶，尿布} → {啤酒}。由于项集{牛奶，尿布，啤酒}的支持度计数是2，而事务的总数是 5，所以规则的支持度为$\frac{2}{5}$=0.4。规则的置信度是项集{牛奶，尿布，啤酒}的支持度计数与项集{牛奶，尿布}支持度计数的商。由于有3 个事务同时包含牛奶和尿布，所以该规则的置信度为$\frac{2}{3}$= 0.67。

1.基本概念

2. 基本定义

- 支持度是一种重要度量，因为支持度很低的规则出现的概率比较低。
- 同样，从商业角度来看，低支持度的规则多半也是无意义的，因为对顾客很少同时购买的商品进行促销可能并无益处。
- 因此，我们想找到的是那些支持度比用户定义的某些阈值大的规则。
- 置信度度量通过规则进行推理的可靠性。
- 对于给定的规则 $X \rightarrow Y$ ，置信度越高，Y在包含X的事务中出现的可能性就越大。
- 置信度也可以估计Y在给定X下的条件概率。

1.基本概念



2. 基本定义

- 关联规则发现：给定事务的集合 T，关联规则发现是指找出支持度大于等于 $minsup$ 并且置信度大于等于 $minconf$ 的所有规则，其中$minsup$和$minconf$是对应的支持度和置信度阈值。
- 挖掘关联规则的一种暴力方法：计算每个可能的规则的支持度和置信度。
- 从数据集中提取的规则的数目可达指数级，所以这种方法的代价太高
  - 假设规则的左边和右边都不是空集，那么从包含 d 个项的数据集提取的可能规则的总数：$R = \sum_{k=1}^{d-1}\left[C_d^k \times \sum_{i=1}^{d-k} C_{d-k}^i\right] = 3^d - 2^{d+1} + 1$
  - 以上表为例， d =6，需要计算$3^6 - 2^7 + 1$ =602条规则的支持度和置信度

1.基本概念

TID	Beer	Bread	Coke	Diaper	Egg	Milk
1	0	1	0	0	0	1
2	1	1	0	1	1	0
3	1	0	1	1	0	1
4	1	1	0	1	0	1
5	0	1	1	1	0	1

2. 基本定义

- 提高关联规则挖掘算法性能的第一步是拆分支持度和置信度要求。
- 规则$X \rightarrow Y$的支持度和其对应项集 $(X \cup Y)$的支持度是一样的。
- 下面的规则有相同的支持度，因为它们涉及的项都源自同一个项集{啤酒，尿布，牛奶}。如果项集{啤酒，尿布，牛奶}是非频繁的，则可以立即剪掉这6个候选规则，而不必计算它们的置信度值。

$$s(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{N}$$

$$c(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}$$

{Milk,Diaper} → {Beer} (s=0.4, c=0.67)
{Milk,Beer} → {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} → {Milk} (s=0.4, c=0.67)
{Beer} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Beer} (s=0.4, c=0.5)
{Milk} → {Diaper,Beer} (s=0.4, c=0.5)

1.基本概念

2. 基本定义

- 大多数关联规则挖掘算法通常采用的一种策略是，将关联规则挖掘任务分解为如下两个主要的子任务：
- 1) 频繁项集产生：其目标是发现满足最小支持度阈值($minsup$)的所有项集。这些项集称作频繁项集(frequent itemset)。
- 2) 规则的产生：其目标是从上一步发现的频繁项集中提取所有高置信度的规则。这些规则称作强规则(strong rule)。
- 频繁项集的产生所需的计算开销远大于规则生成所需的计算开销。

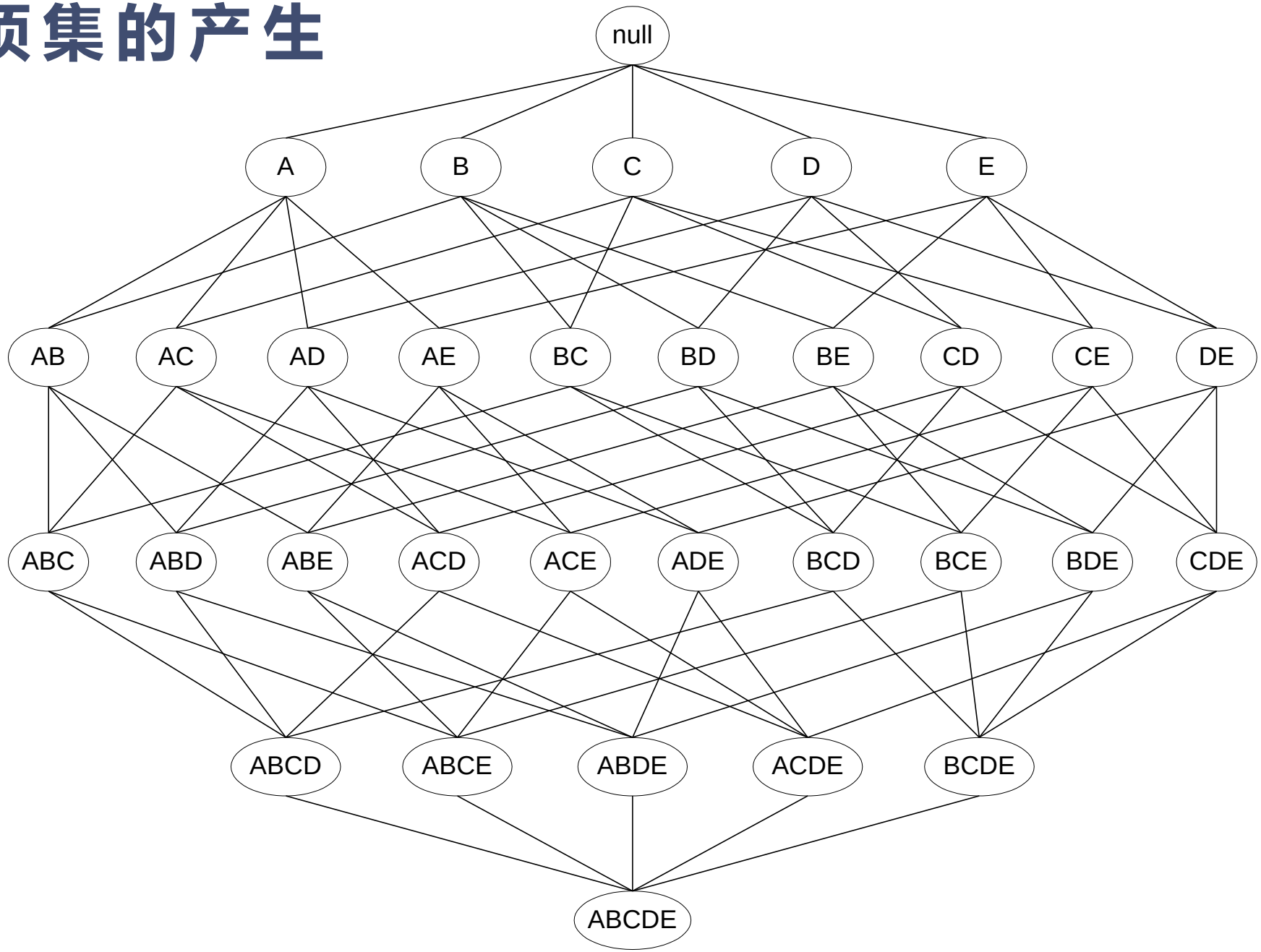
目 录

2.频繁项集的产生

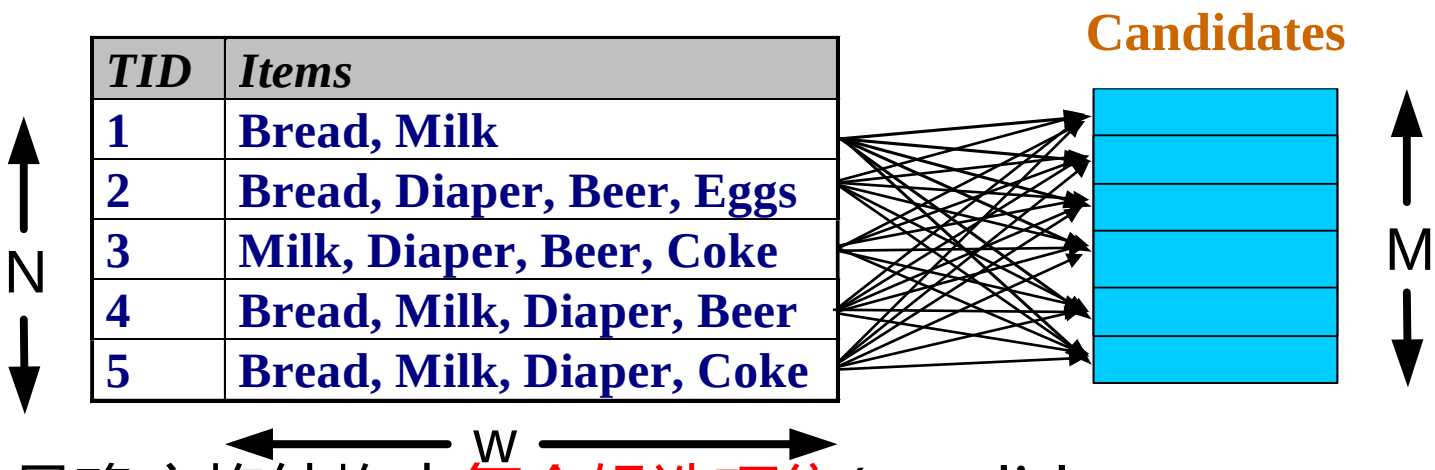
1. 基本思路

- 格结构(lattice structure)常常用于枚举所有可能的项集。
- 一般来说，一个包含k个项的数据集可能产生不包括空集在内的(2^k-1)个频繁项集。
- 由于在许多实际应用中k的值可能非常大，因此需要探查的项集搜索空间可能是指数规模的。

2.频繁项集的产生


null
A
B
C
D
E
AB
AC
AD
AE
BC
BD
BE
CD
CE
DE
ABC
ABD
ABE
ACD
ACE
ADE
BCD
BCE
BDE
CDE
ABCD
ABCE
ABDE
ACDE
BCDE
ABCDE

2.频繁项集的产生



1. 基本思路

- 发现频繁项集的一种暴力方法是确定格结构中每个候选项集(candidate itemset)的支持度计数。
- 为了完成这一任务，必须将每个候选项集与每个事务进行比较。
- 如果候选项集包含在事务中，则候选项集的支持度计数会增加。
  - 例如，由于项集{面包，牛奶}出现在事务1、4和5中，其支持度计数将增加3次。
- 这种方法的开销可能非常大，因为它需要进行$O(NMw)$次比较，其中N是事务数，$M = 2^k - 1$是候选项集数，而w是事务的最大宽度 。

2.频繁项集的产生

1. 基本思路

- 降低产生频繁项集的计算复杂度有 3 种主要方法:
- 1) 减少候选项集的数目(M)。先验(apriori)原理是一种不用计算支持度值而删除某些候选项集的有效方法。
- 2) 减少比较次数(MN)。不用将每个候选项集与每个事务相匹配，而是选择使用更高级的数据结构来存储候选项集，以此减少比较次数。
- 3) 减少事务数目(N)。随着候选项集的规模越来越大，能支持项集的事务将越来越少。例如:在搜索大小为3 或更大的频繁项集之前将宽度为 2 事务去掉。

2.频繁项集的产生

2. 先验原理

- **定理5.1 先验原理**：如果一个项集是频繁的，则它的所有子集一定也是频繁的。

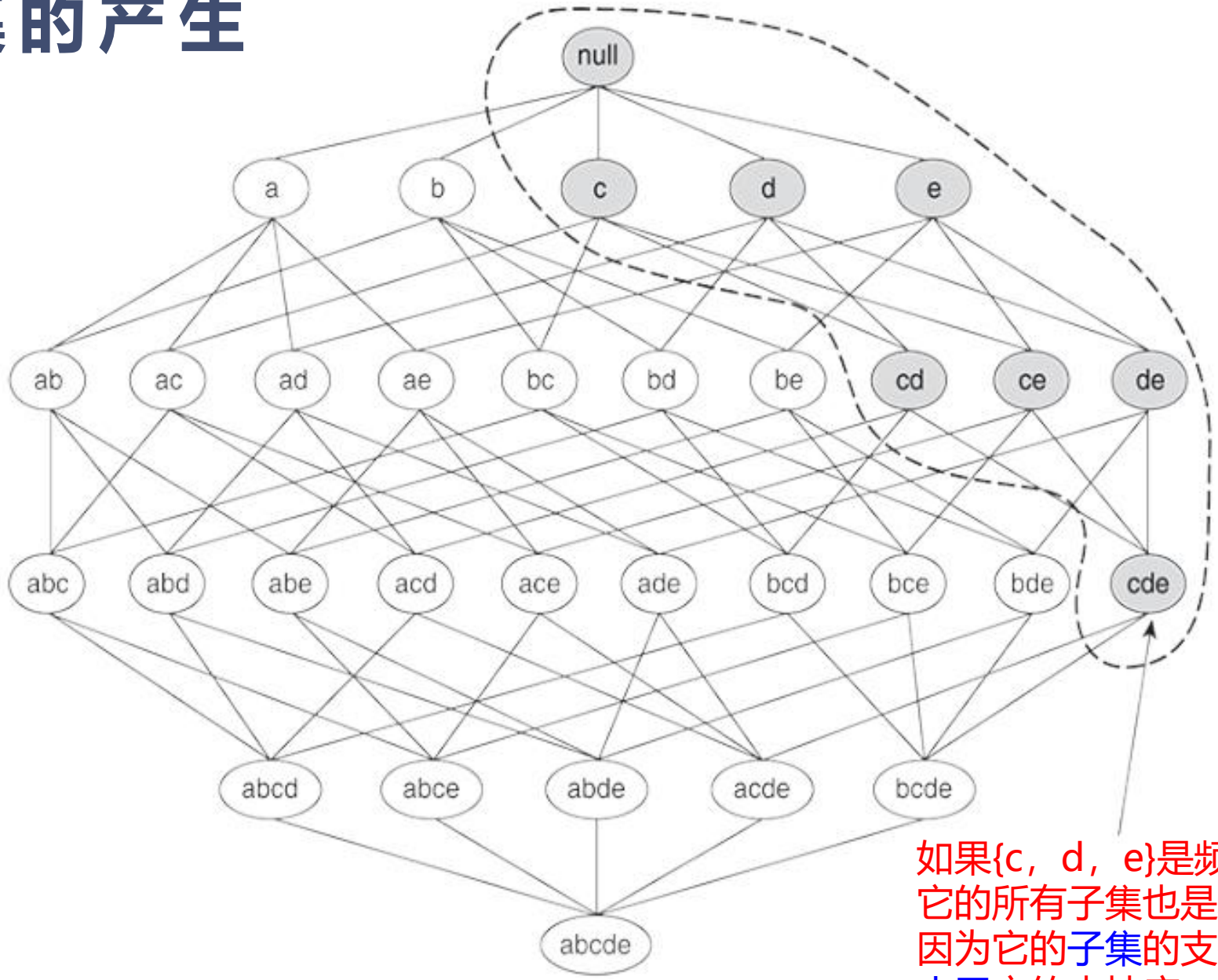
如果Y是频繁的，则Y的子集X一定也是频繁的
如果X是非频繁的，则X的超集Y一定也是非频繁的

- 先验原理适用于以下的支持度的属性：

$$\forall X, Y: (X \subseteq Y) \Longrightarrow s(X) \geq s(Y)$$

- 一个项集的支持度永远不会超过其子集的支持度。
- 这个性质也称为支持度度量的反单调性(anti-monotone)。
- 如果对于项集Y的每个子集X(即$X \subseteq Y$)，有 $f(Y) \leq f(X)$，那么度量f具有反单调性。

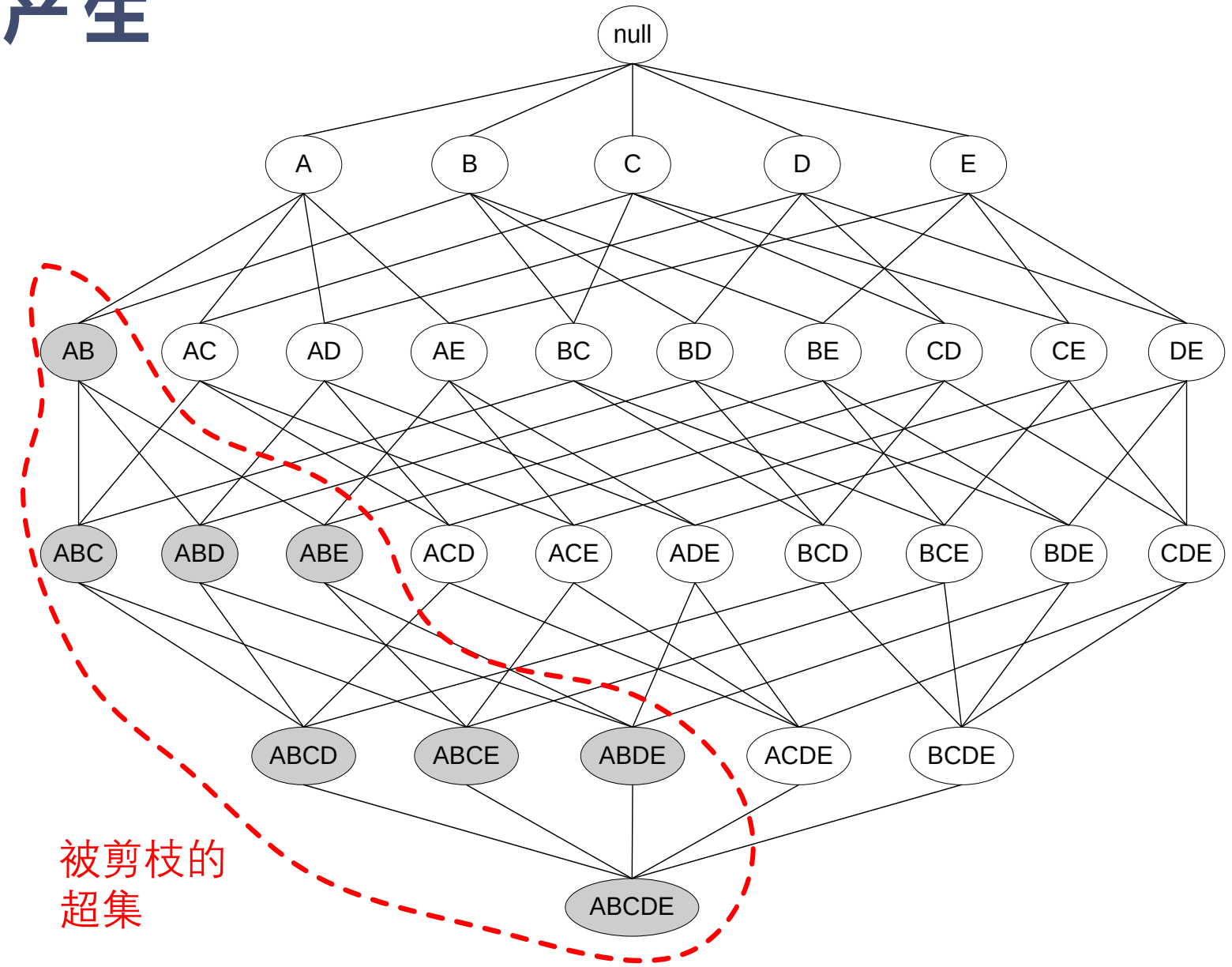
2.频繁项集的产生



如果{c，d，e}是频繁的，则它的所有子集也是频繁的，因为它的子集的支持度都不小于它的支持度

2.频繁项集的产生

如果{A，B}是非频繁的，则它的所有超集也是非频繁的，因为它的所有超集的支持度都不大于它的支持度



2.频繁项集的产生

3. Apriori算法的频繁项集产生

- Apriori算法是第一个关联规则挖掘算法，它开创性地使用基于支持度的剪枝技术，系统地控制候选项集的指数增长 。
- 初始时，每个项都被看作候选 1-项集。对它们的支持度计数之后，候选项集{可乐}和{鸡蛋}被丢弃。

TID	*Items*
1	Bread, Milk
2	Beer, Bread, Diaper, Eggs
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk

候选 (1-项集)

Item	Count
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Minimum Support = 3

频繁 (1-项集)

Item	Count
Bread	4
Milk	4
Beer	3
Diaper	4

2.频繁项集的产生

TID	Items
1	Bread, Milk
2	Beer, Bread, Diaper, Eggs
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk

3. Apriori算法的频繁项集产生

- 由于有4个频繁1-项集，因此算法产生的候选 2-项集的数目为C_4^2。计算它们的支持度值之后，发现这6个候选项集中的{啤酒，面包}和{啤酒，牛奶}是非频繁的。

Minimum Support = 3

频繁 (1-项集)

Item	Count
Bread	4
Milk	4
Beer	3
Diaper	4

候选 (2-项集)

Itemset	Count
{Bread,Milk}	3
{Beer, Bread}	2
{Bread,Diaper}	3
{Beer,Milk}	2
{Diaper,Milk}	3
{Beer,Diaper}	3

频繁 (2-项集)

Itemset	Count
{Bread,Milk}	3
{Bread,Diaper}	3
{Diaper,Milk}	3
{Beer,Diaper}	3

2.频繁项集的产生

TID	Items
1	Bread, Milk
2	Beer, Bread, Diaper, Eggs
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk

3. Apriori算法的频繁项集产生

- 剩下的4个候选 2-项集是频繁的，因此用来产生候选 3-项集。只需要保留其子集都频繁的候选 3-项集。唯一具有这种性质的候选是{面包，尿布，牛奶}。然而，尽管{面包，尿布，牛奶}的子集是频繁的，但该项集本身并非如此。

Minimum Support = 3

频繁 (2-项集)

Itemset	Count
{Bread,Milk}	3
{Bread,Diaper}	3
{Diaper,Milk}	3
{Beer,Diaper}	3

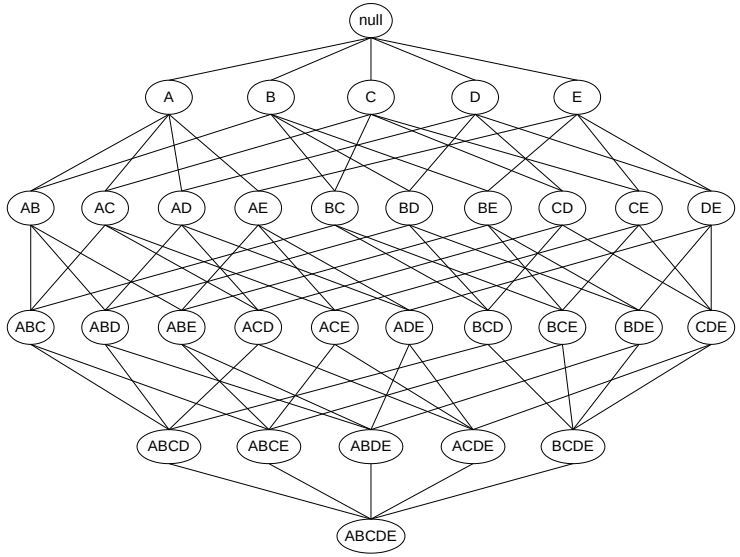
候选 (3-项集)

Itemset	Count
{Bread,Diaper,Milk}	2

2.频繁项集的产生

算法5.1 Apriori算法的频繁项集产生	
1:	$k = 1$
2:	$F_k = \{i \mid i \in I \wedge \sigma(\{i\}) \geq N \times minsup\}$ /*找出所有的频繁1-项集*/
3:	**repeat**
4:	$k = k + 1$
5:	$C_k = candidate - gen(F_{k-1})$ /*产生候选项集*/
6:	$C_k = candidate - prune(C_k, F_{k-1})$ /*剪枝候选项集*/
7:	**for each** $t \in T$ **do**
8:	$C_t = subset(C_k, t)$ /*识别所有属于t的候选$C_t \subseteq C_k$*/
9:	**for each** $c \in C_t$ **do**
10:	$\sigma(c) = \sigma(c) + 1$ /*支持度计数增加*/
11:	**end for**
12:	**end for**
13:	$F_k = \{c \mid c \in C_k \wedge \sigma(c) \geq N \times minsup\}$ /*提取频繁 k-项集*/
14:	**until** $F_k = \emptyset$
15:	**Result** $= \cup F_k$

2.频繁项集的产生



3. Apriori算法的频繁项集产生

- Apriori算法的频繁项集产生的部分有两个重要的特点:
- 第一，它是一个逐层(level-wise)算法，即从频繁 1-项集到最长的频繁项集，它每次遍历项集格中的一层;
- 第二，它使用产生-测试(generate-and-test)策略来发现频繁项集。在每次迭代之后，新的候选项集都由前一次迭代发现的频繁项集产生，然后对每个候选项集的支持度进行计数，并与最小支持度阈值进行比较。该算法需要的总迭代次数是$k_{max}+1$，其中，k_{max}是频繁项集的最大长度。

2.频繁项集的产生

4. 候选项集的产生与剪枝

- 1)候选项集的产生。该操作由前一次迭代发现的频繁$(k-1)$-项集产生新的候选 k-项集。
- 2)候选项集的剪枝。该操作采用基于支持度的剪枝策略，删除一些候选 k-项集。也就是说，如果某一k-项集的子集在之前迭代中被发现是非频繁的，则将这一 k-项集删除。需要注意的是，完成这种剪枝并不需要计算这些 k-项集的实际支持度(支持度可以通过将这些项集和每个事务进行比较获得)。

2.频繁项集的产生

4. 候选项集的产生与剪枝 --- 候选项集的产生

- 如果候选项集产生过程中未遗漏任何频繁项集，则称其为完备的(complete)。
- 为确保完备性，候选项集的集合必须包含所有频繁项集的集合，即 $\forall k: F_k \subseteq C_k$。
- 如果候选项集产生过程对同一候选项集的生成不超过一次，则称之为无冗余的(non-redundant)。

2.频繁项集的产生

4. 候选项集的产生与剪枝 --- 候选项集的产生

- $F_{k-1} \times F_{k-1}$方法：Apriori算法使用的函数 candidate-gen 的候选产生过程是将一对频繁$(k-1)$-项集进行合并，仅当其前$(k-2)$个项按照字典序都相同。
- 令$A=\{a_1, a_2, \cdots, a_{k-1}\}$和$B=\{b_1, b_2, \cdots, b_{k-1}\}$是一对符合字典序的频繁$(k-1)$-项集，合并$A$和$B$，如果它们满足如下条件:

$$a_i = b_i (i = 1,2, \ldots, k-2)$$

- 注意在这种情况下，$a_{k-1} \neq b_{k-1}$，因为A和B是两个不一样的项集。

2.频繁项集的产生

$A = \{a_1, a_2, \cdots, a_{k-1}\}$　　$B = \{b_1, b_2, \cdots, b_{k-1}\}$

$\{a_1, a_2, \cdots, a_{k-1}, b_{k-1}\}$

4. 候选项集的产生与剪枝 --- 候选项集的产生

- 通过合并A和B生成的候选k-项集包括相同的前$(k-2)$个项，紧接着是按照字典序的a_{k-1}和b_{k-1}。这个候选产生过程是完备的，因为对于每个符合字典序的频繁 k-项集，都存在着两个按字典序的频繁$(k-1)$-项集，它们在前$(k-2)$个位置都包含了相同的项。
- 使用字典序可以避免重复生成，因而是无冗余的。
- 比如：3-项集 {面包，尿布，牛奶}是由两个按字典序的频繁2-项集{面包，尿布}和{面包，牛奶}合并产生的。它们的第一个位置包含了相同的项{面包}。

2.频繁项集的产生

4. 候选项集的产生与剪枝 ---候选项集的产生

算法：$candidate-gen(F_{k-1})$	
1:	$C_k = \emptyset$
2:	**for each** $f_1 \in F_{k-1}$
3:	**for each** $f_2 \in F_{k-1}$
4:	**if** $(f_1[1] = f_2[1]) \wedge (f_1[2] = f_2[2]) \wedge \cdots \wedge (f_1[k-2] = f_2[k-2]) \wedge (f_1[k-1] < f_2[k-1])$ **then**
5:	$c = \{f_1[1], f_1[2], \cdots, f_1[k-2], f_1[k-1], f_2[k-1]\}$ /*合并f_1与f_2*/
6:	$C_k = C_k \cup c$ /*将新生成的候选项集c加入到C_k中*/
7:	**end if**
8:	**end for**
9:	**end for**
10:	**Return** C_k

2.频繁项集的产生

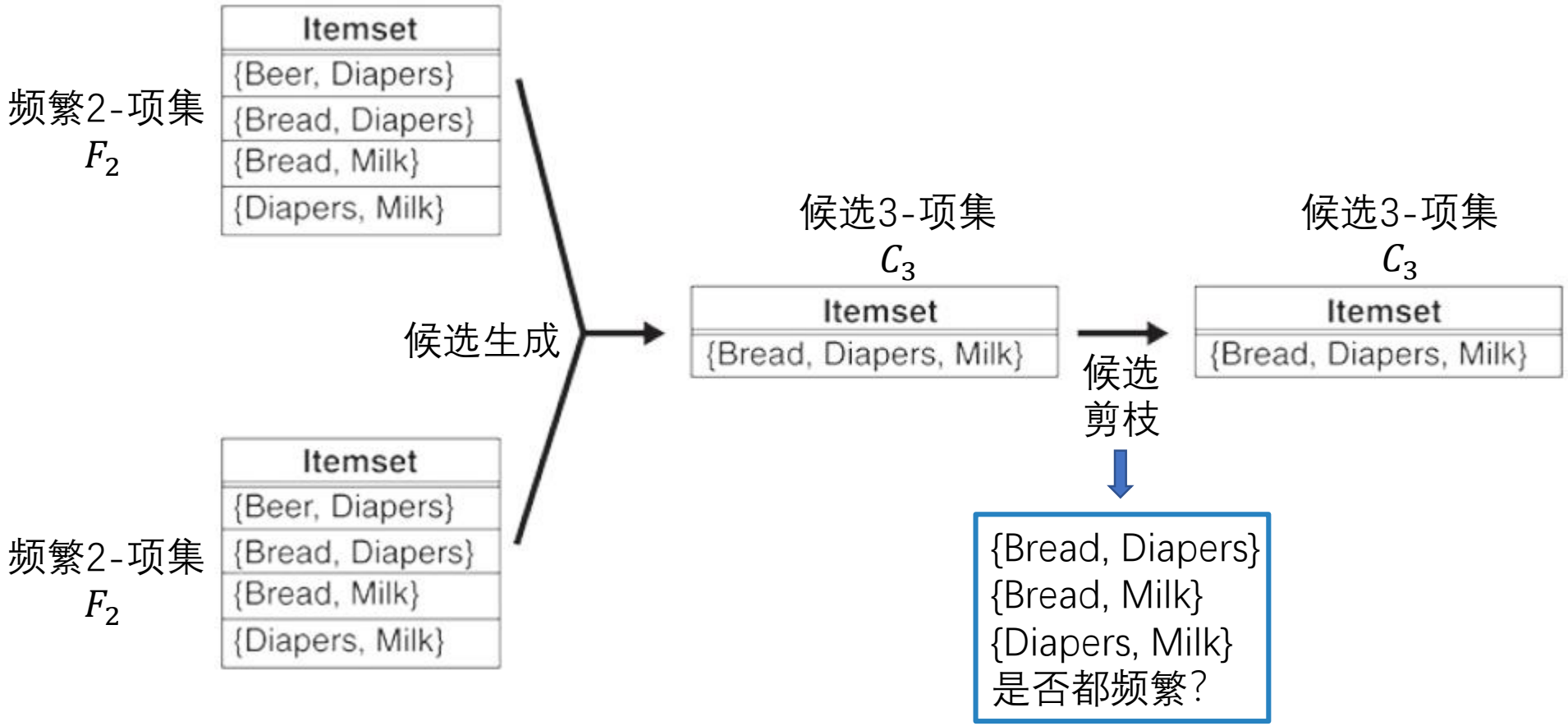
4. 候选项集的产生与剪枝 --- 候选项集的剪枝

■ 按照Apriori原理，只要X的真子集中有任何一个是非频繁的，X会被立即剪枝。

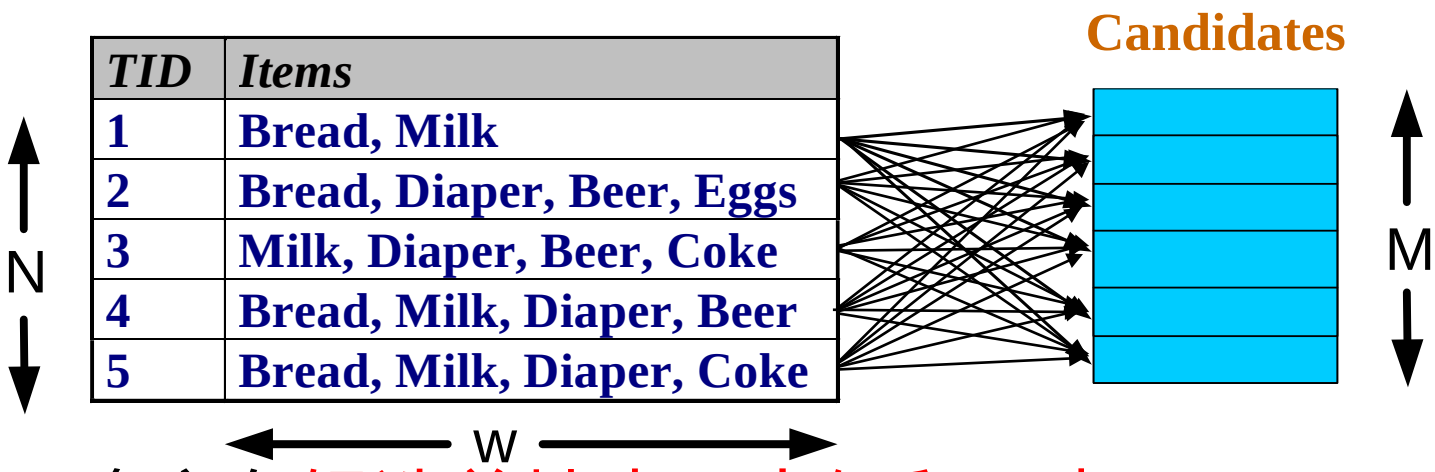
算法：$candidate-prune(C_k, F_{k-1})$	
1:	for each $c \in C_k$
2:	for each $(k-1)-$subset s of c
3:	if $s \notin F_{k-1}$ then /*如果c的某一个真子集s是非频繁的*/
4:	$C_k = C_k - c$ /*将c从候选项集C_k中移除*/
5:	break
6:	end if
7:	end for
8:	end for
9:	Return C_k

2.频繁项集的产生

4. 候选项集的产生与剪枝



2.频繁项集的产生



5. 支持度计数

- 支持度计数过程的作用是，确定在候选剪枝步骤中保留下来的每个候选项集出现的频繁程度。
- 支持度计数在算法5.1的第7-12行实现。
- 支持度计数的一种暴力方法是，将每个事务与所有的候选项集进行比较，并更新包含在事务中的候选项集的支持度计数。这种方法的计算开销十分高($O(NMw)$)，尤其当事务和候选项集的数目都很大时更为明显。

2.频繁项集的产生

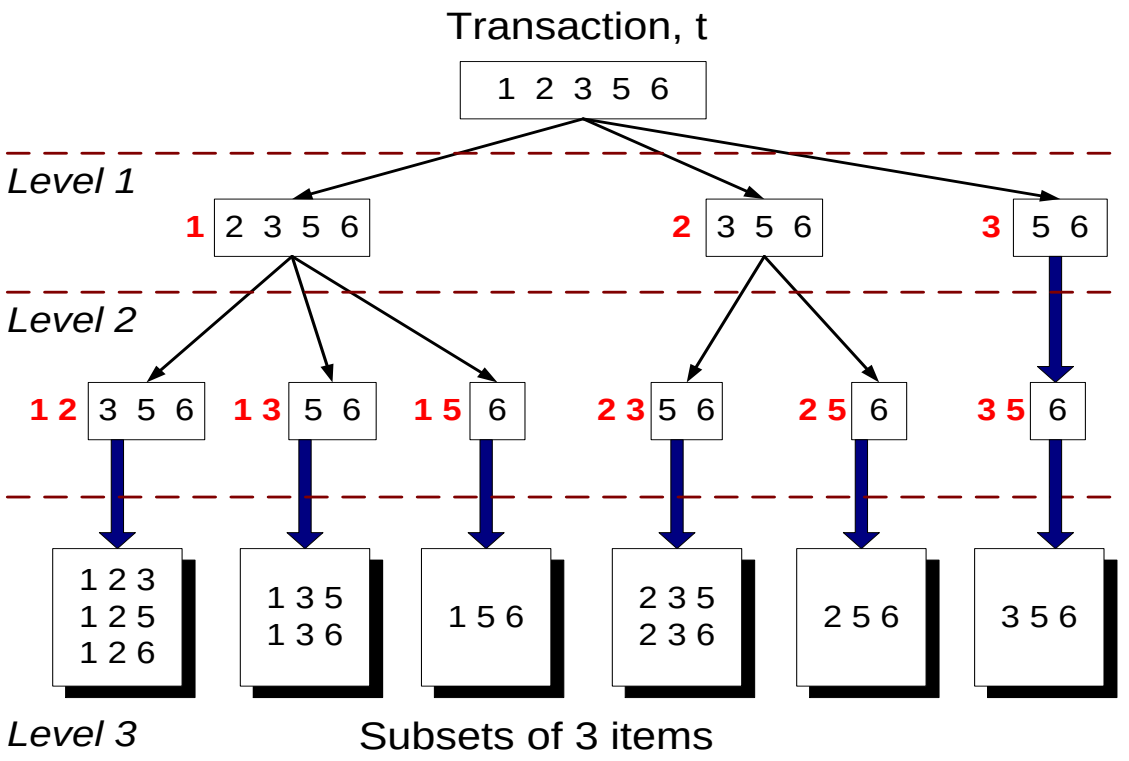
5. 支持度计数 --- 枚举事务所包含的项集

- 另一种方法是枚举每个事务所包含的项集，并且利用其更新对应的候选项集的支持度。
- 例如，考虑事务t，它包含5个项{1，2，3，5，6}。该事务包含C_5^3=10个大小为3的项集，其中的某些项集可能对应于所考察的候选3-项集，遇到这种情况则增加其支持度。而那些不与任何候选项集对应的事务t的子集则可以忽略。

2.频繁项集的产生

5. 支持度计数 --- 枚举事务所包含的项集

- 假定每个项集中的项都以递增的字典序排列，则项集可以这样枚举：先指定最小项，其后跟随较大的项。
- 例如，给定t={1，2，3，5，6}，它的所有3-项集一定以项1、2或3开始。不能构造以5或6开始的3-项集，因为事务t中只有两个项的标号大于等于5。



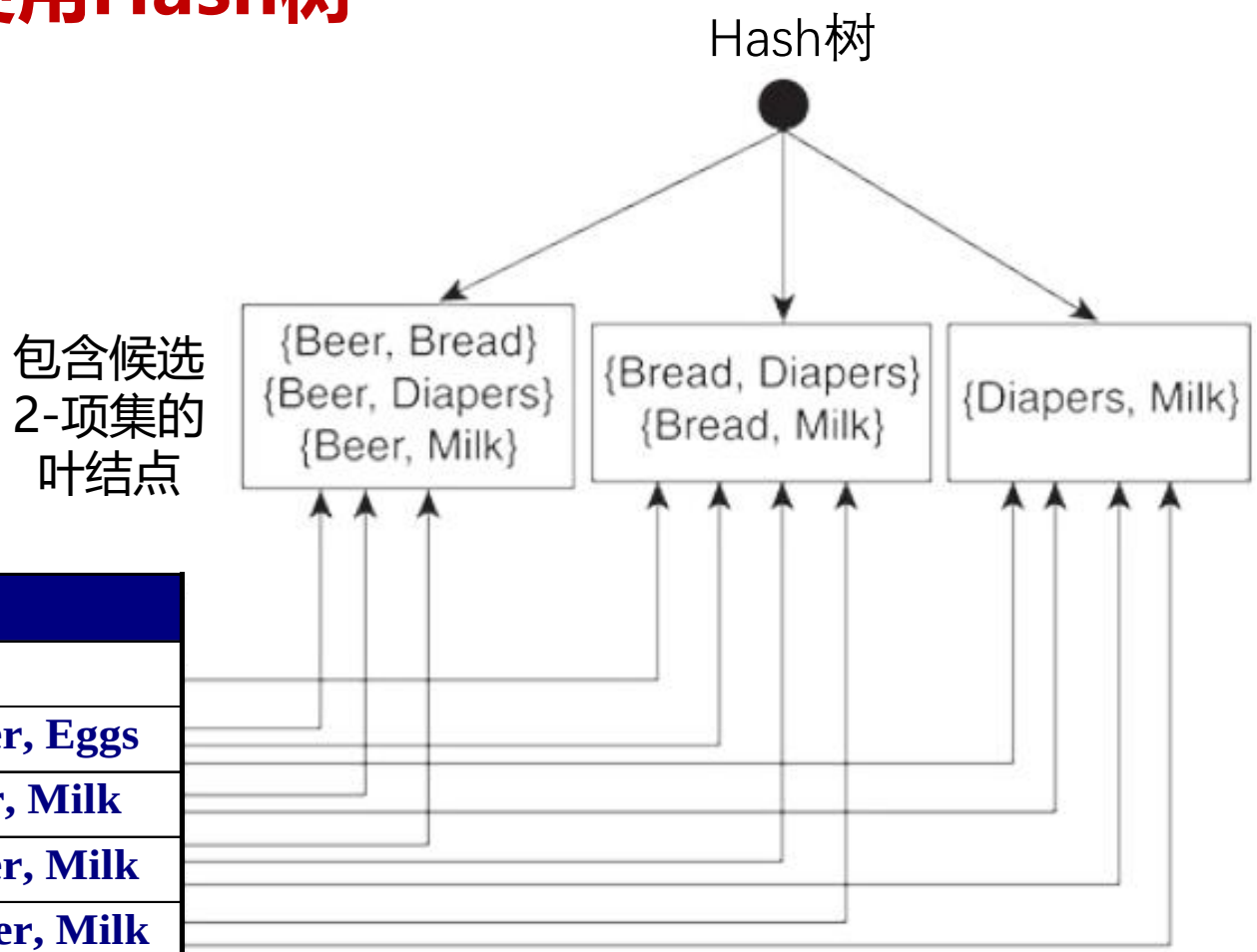
2.频繁项集的产生

5. 支持度计数 --- 使用Hash树

- 接着还必须确定每一个枚举的3-项集是否对应于一个候选项集，如果它与一个候选项集匹配，则相应候选项集的支持度计数增加。
- 在Apriori算法中，候选项集被划分为不同的桶，并存放在Hash树中。
- 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。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的候选项集进行匹配。

2.频繁项集的产生

5. 支持度计数 --- 使用Hash树

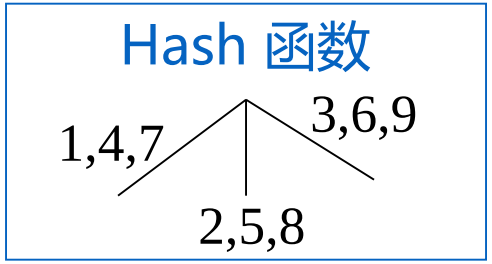


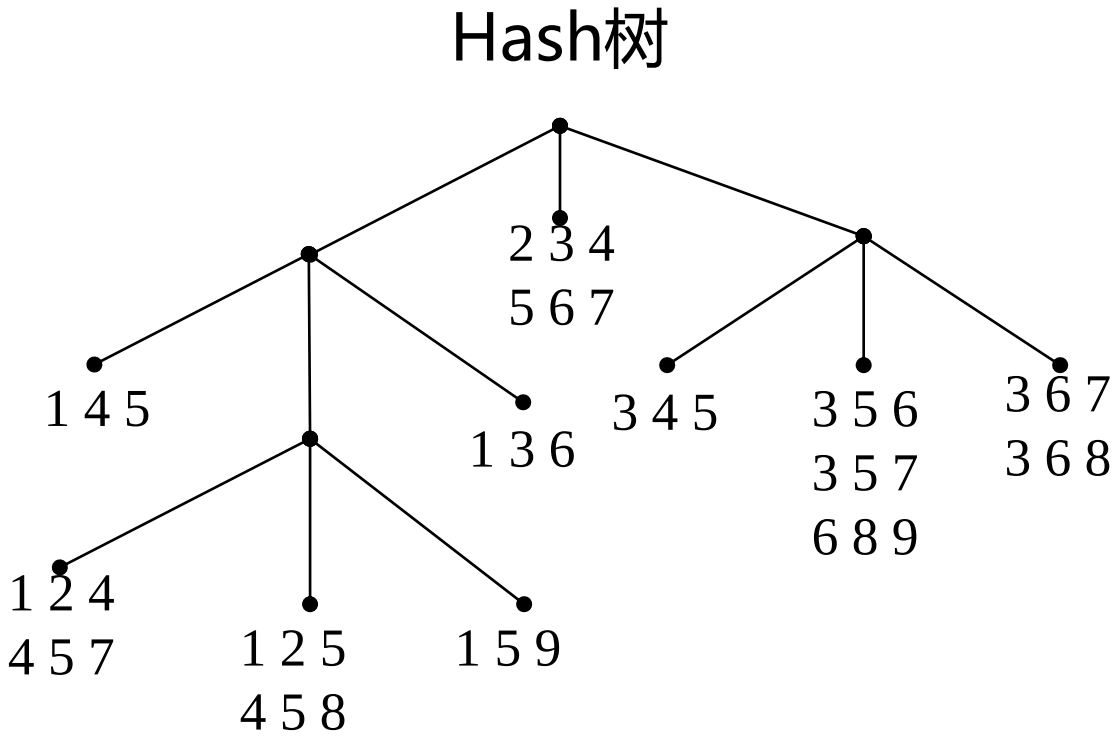
TID	*Items*
1	Bread, Milk
2	Beer, Bread, Diaper, Eggs
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk

2.频繁项集的产生

5. 支持度计数 --- 使用Hash树

- 假设有15个长度为3的候选项目集:
- {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}
- 所有的候选项集都存放在Hash树的叶结点中

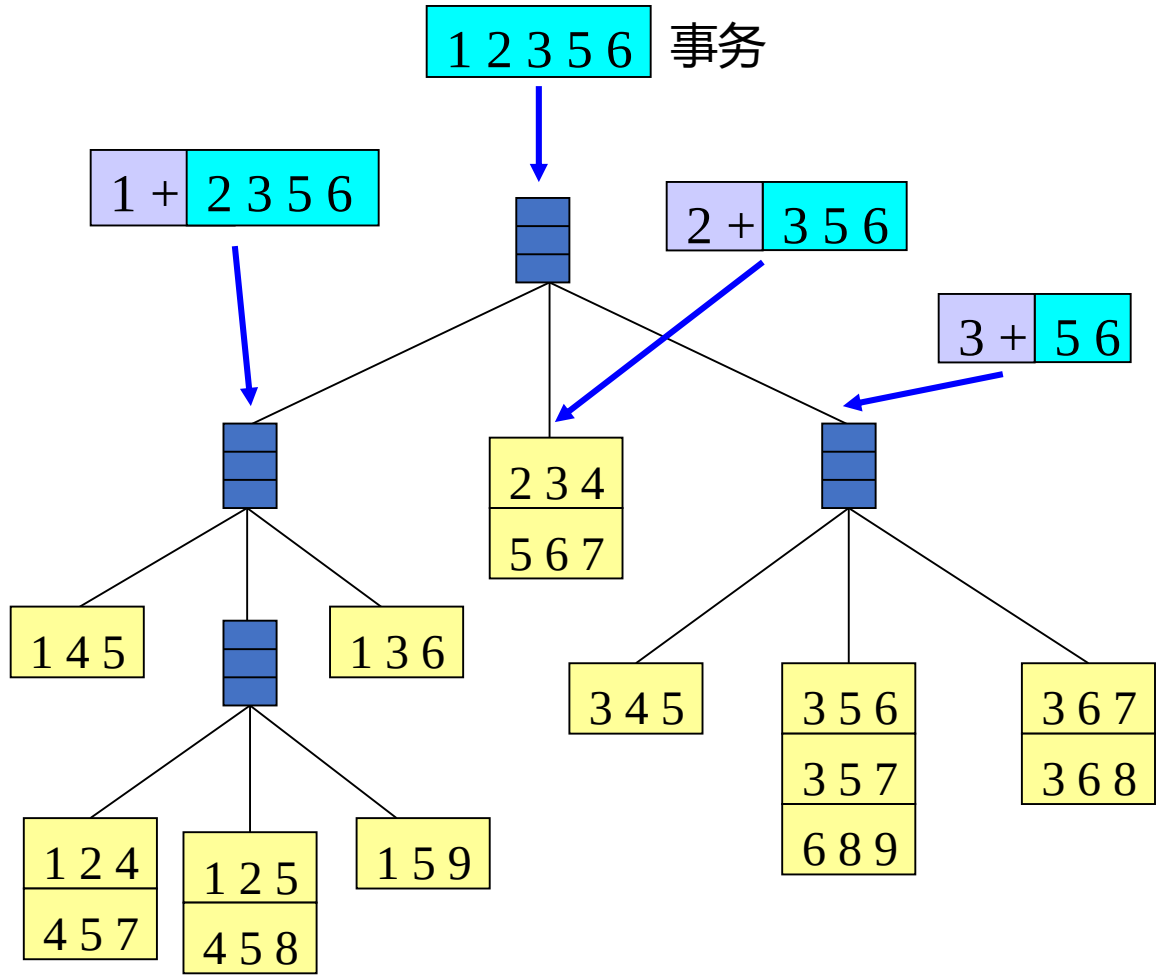


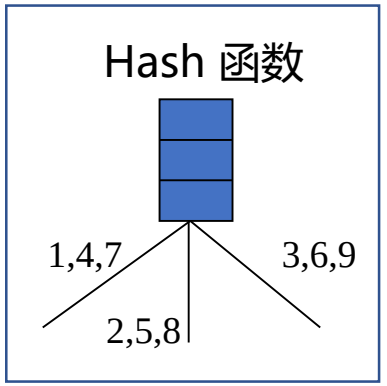


2.频繁项集的产生

5. 支持度计数 --- 使用Hash树

事务中的项1、2和3将分别散列。项1被散列到根结点的左子树，项2被散列到中间子树，而项3被散列到右子树。

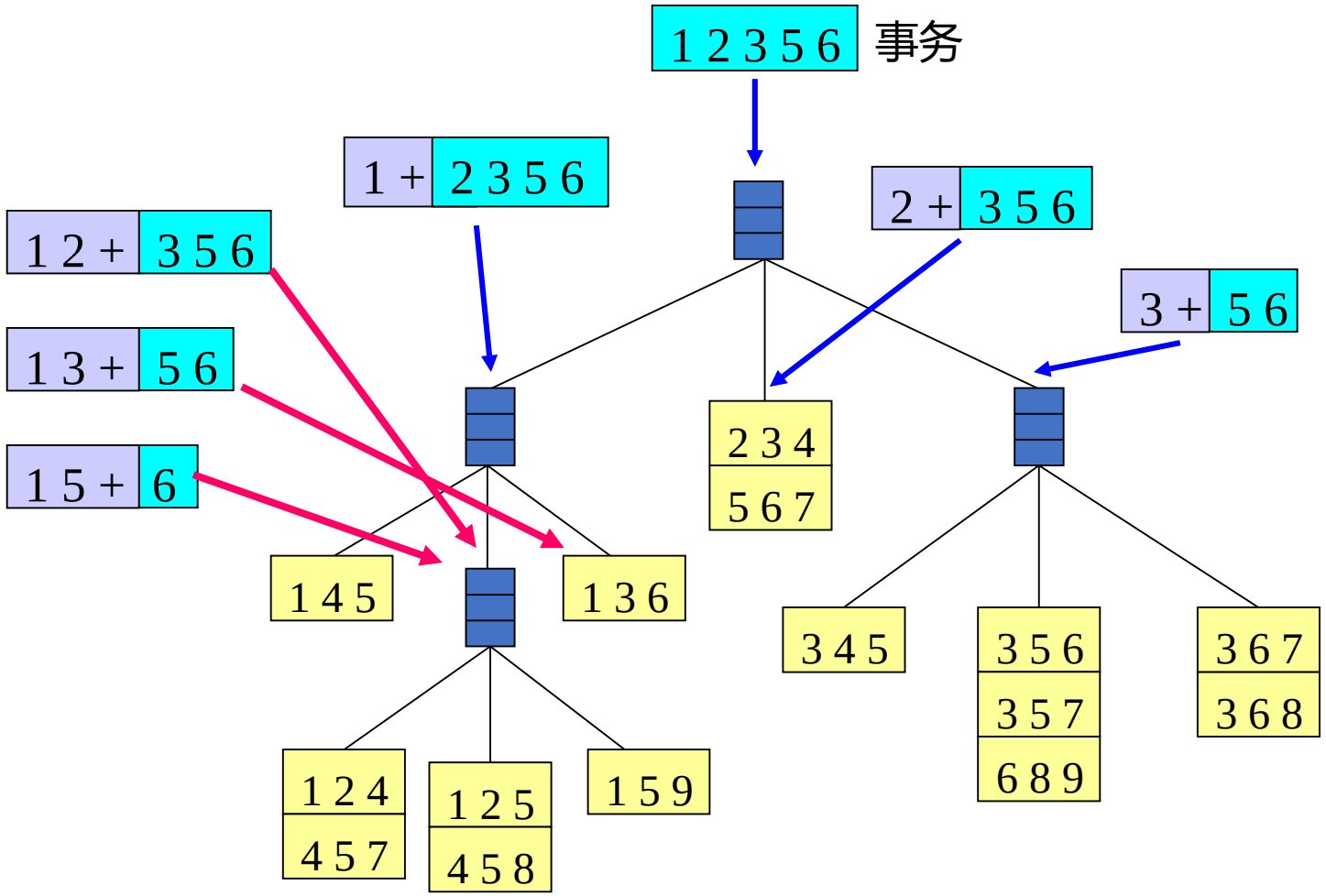


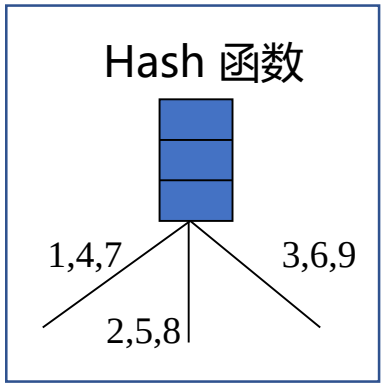


2.频繁项集的产生

5. 支持度计数 --- 使用Hash树

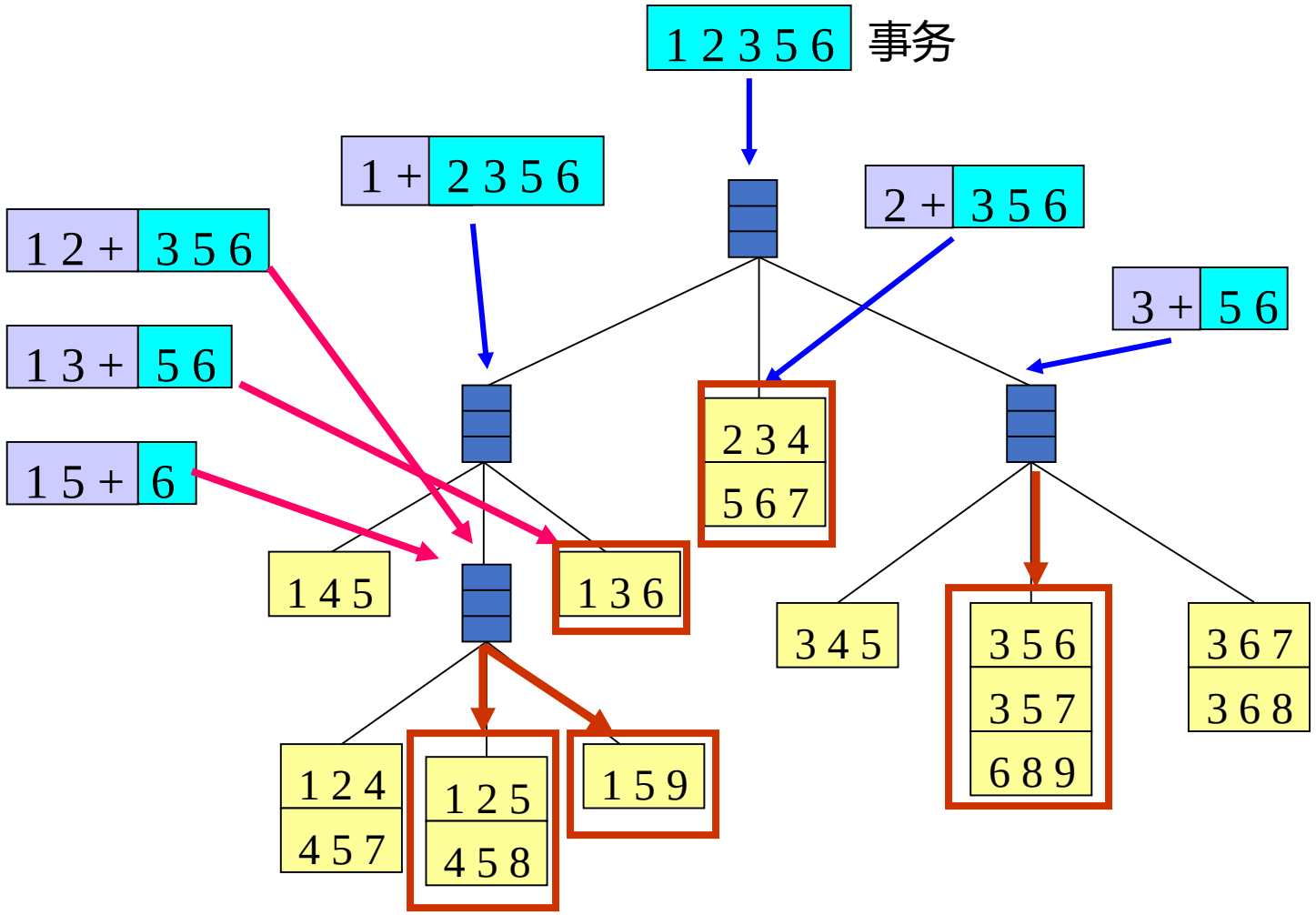
在根结点散列项1之后，散列事务的项2、3和5。按照Hash函数，项2和项5散列到中间子树，而项3散列到右子树。

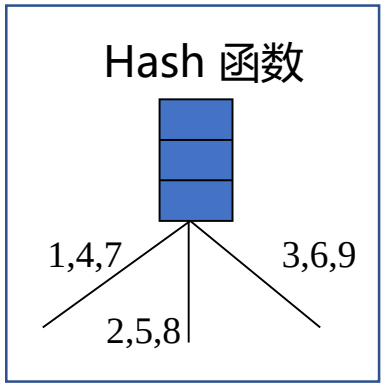




2.频繁项集的产生

5. 支持度计数 --- 使用Hash树





继续该过程，直至到达Hash树的叶结点。将存放在被访问过的叶结点中的候选项集与事务进行比较，如果候选项集是该事务的子集，则增加它的支持度计数。

在这个例子中，访问了9个叶结点中的5个，同时将15个项集中的9个与事务进行了比较。

目 录

3.规则的产生

$$s(A \to B) = \frac{\sigma(A \cup B)}{N} \qquad c(A \to B) = \frac{\sigma(A \cup B)}{\sigma(A)}$$

1. 基本概念

- 忽略前因或后果为空的规则（$\emptyset \to Y$或$Y \to \emptyset$），每个频繁k-项集Y最多可以产生$(2^k - 2)$个关联规则。--- 等于Y的非空真子集的个数
- 关联规则可以这样提取：将项集Y划分成两个非空的子集X和$Y - X$，使得$X \to Y - X$满足置信度阈值。这样的规则必然已经满足支持度阈值，因为它们是由频繁项集产生的。
- 计算关联规则的置信度并不需要再次扫描事务数据集，因为X和$Y - X$这两个项集的支持度计数已经在频繁项集产生时得到。

3.规则的产生

$$s(A \to B) = \frac{\sigma(A \cup B)}{N} \qquad c(A \to B) = \frac{\sigma(A \cup B)}{\sigma(A)}$$

1. 基本概念

- 设$Y = \{a, b, c\}$是频繁项集。可以由Y产生**6**个候选关联规则：$\{a, b\} \to \{c\}$，$\{a, c\} \to \{b\}$，$\{b, c\} \to \{a\}$，$\{a\} \to \{b, c\}$，$\{b\} \to \{a, c\}$和$\{c\} \to \{a, b\}$。由于它们的支持度都等于Y的支持度，这些规则一定满足支持度阈值。
- 对于规则$\{a, b\} \to \{c\}$，该规则的置信度c为

$$c = \frac{\sigma(\{a, b, c\})}{\sigma(\{a, b\})}$$

- 因为$\{a, b\}$是$\{a, b, c\}$的子集，并且$\{a, b, c\}$是频繁项集，所以它们的支持度计数都是已知的。

3.规则的产生

$$s(A \to B) = \frac{\sigma(A \cup B)}{N} \qquad c(A \to B) = \frac{\sigma(A \cup B)}{\sigma(A)}$$

2. 基于置信度的剪枝

- 定理5.2：令Y是一个项集，X是Y的一个子集。如果规则$X \to Y - X$不满足置信度阈值，则形如$\tilde{X} \to Y - \tilde{X}$的规则一定也不满足置信度阈值，其中$\tilde{X}$是$X$的子集（$\tilde{X} \subseteq X$）。
- 规则$\tilde{X} \to Y - \tilde{X}$和$X \to Y - X$的置信度分别是：$\frac{\sigma(Y)}{\sigma(\tilde{X})}$和$\frac{\sigma(Y)}{\sigma(X)}$。由于$\tilde{X} \subseteq X$，所以$\sigma(\tilde{X}) \geq \sigma(X)$。因此，$\frac{\sigma(Y)}{\sigma(\tilde{X})} \leq \frac{\sigma(Y)}{\sigma(X)}$，即$c(\tilde{X} \to Y - \tilde{X}) \leq c(X \to Y - X)$。
- 注意：由于$\tilde{X} \subseteq X$，$(Y - X) \subseteq (Y - \tilde{X})$，即前者是后者的子集。

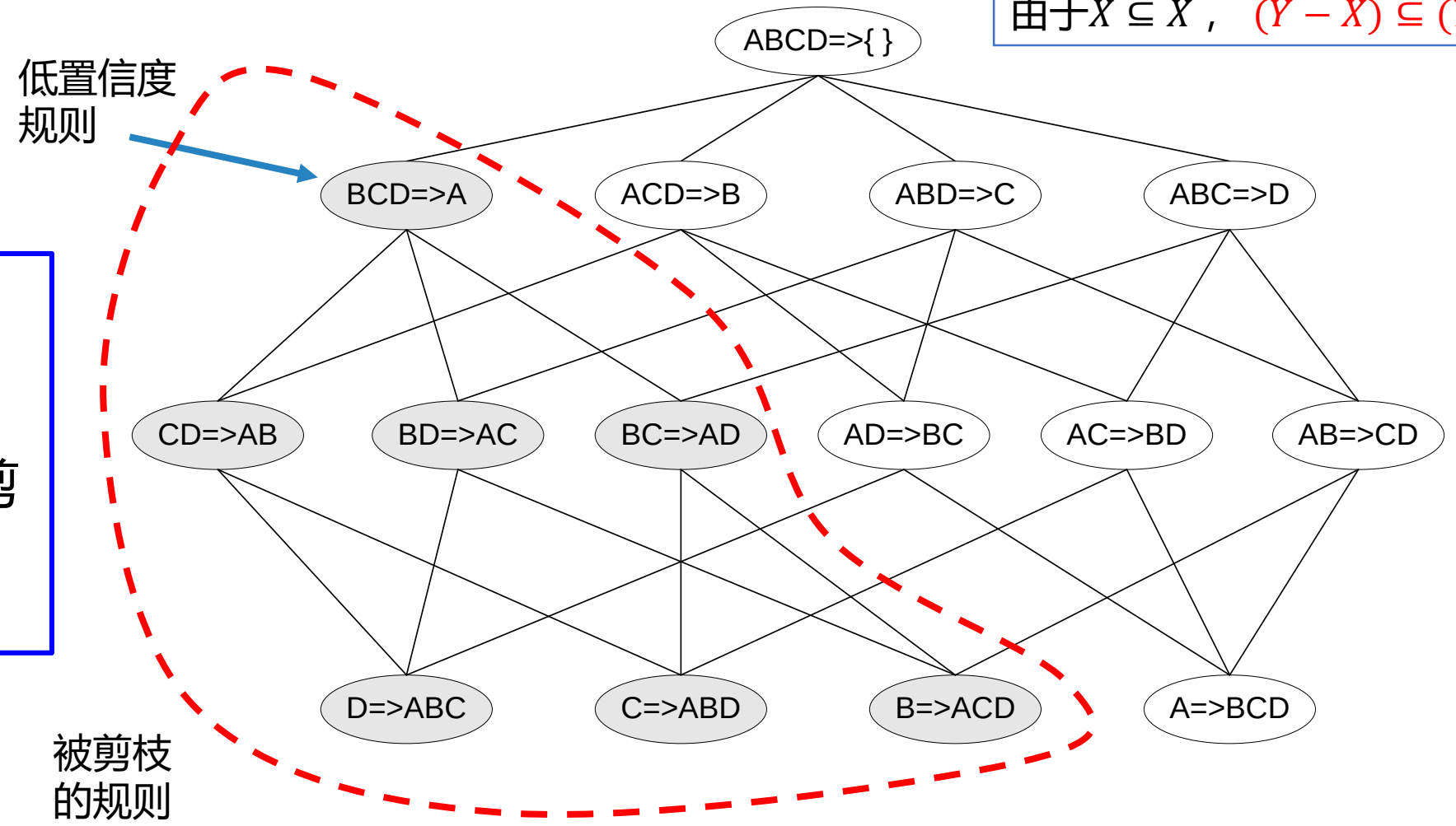
3. Apriori算法中规则的产生

- Apriori算法使用一种逐层方法来产生关联规则，其中每层对应于规则后果中的项数。
- 首先，提取规则后果只含一个项的所有高置信度规则，然后，使用这些规则来产生新的候选规则。
- 例如，如果$\{acd\} \rightarrow \{b\}$和$\{abd\} \rightarrow \{c\}$是两个高置信度规则，则通过合并这两个规则的后果产生候选规则$\{ad\} \rightarrow \{bc\}$。

3.规则的产生

定理 5.2：$\tilde{X}$是X的子集，如果规则 $X \rightarrow Y - X$不满足置信度阈值，则形如$\tilde{X} \rightarrow Y - \tilde{X}(\tilde{X} \subseteq X)$的规则一定也不满足。

由于$\tilde{X} \subseteq X$，$(Y - X) \subseteq (Y - \tilde{X})$



如果格中的任意结点具有低置信度，则根据定理5.2，可以立即剪掉该结点生成的整个子图。

由频繁项集$\{a, b, c, d\}$产生关联规则的格结构

3. Apriori算法中规则的产生

- 对于每一个频繁k-项集f_k，生成相关的规则
- 例如：对于频繁项集$f_4 = \{a, b, c, d\}$，$H_1 = \{\{a\}, \{b\}, \{c\}, \{d\}\}$

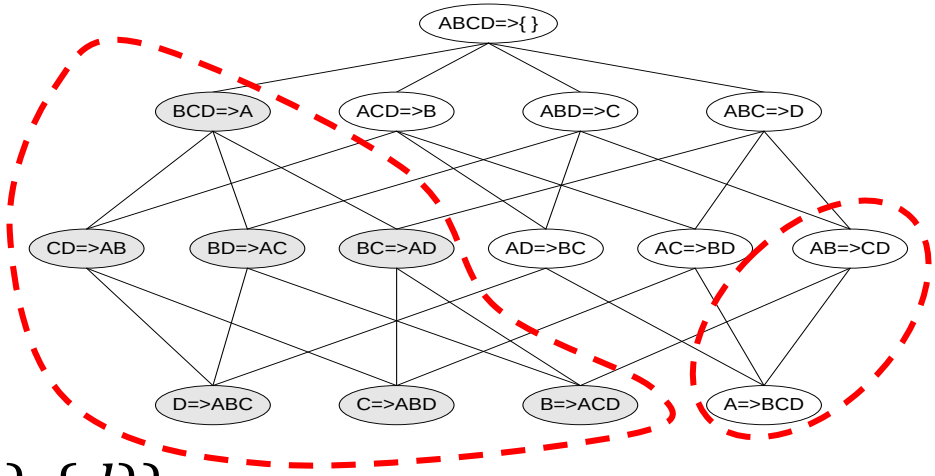
算法5.2 Apriori算法中关联规则的生成	
1:	for each 频繁k-项集f_k，$k \geq 2$ do
2:	$H_1 = \{i \mid i \in f_k\}$ /*规则的1-项后果*/
3:	$ap-genrules(f_k, H_1, k, 1)$
4:	end for

3.规则的产生

算法5.3 $ap-genrules(f_k，H_m，k，m)$过程	
1:	if $(k \geq m+1)$ and $(H_m \neq \emptyset)$ then
2:	for each $h_m \in H_m$ do
3:	$conf = \sigma(f_k)/\sigma(f_k - h_m)$ /*计算规则的置信度*/
4:	if $conf \geq minconf$ then
5:	output $(f_k - h_m) \rightarrow h_m$ /*输出规则*/
6:	else
7:	$H_m = H_m - h_m$ /*将h_m从H_m中移除*/
8:	end if
9:	end for
10:	$H_{m+1} = candidate-gen(H_m)$ /*生成长度为$m+1$的项集*/
11:	$H_{m+1} = candidate-prune(H_{m+1}, H_m)$ /*进行剪枝*/
12:	$ap-genrules(f_k, H_{m+1}, k, m+1)$
13:	end if

3.规则的产生

红色虚线框里的是不满足置信度阈值的候选规则



3. Apriori算法中规则的产生

- 对于频繁项集$f_4 = \{a, b, c, d\}$，$H_1 = \{\{a\}, \{b\}, \{c\}, \{d\}\}$
- 可以生成4条候选规则：bcd→a，acd→b，abd→c，abc→d
- 假设第1条候选规则的置信度小于阈值，$H_1 = \{\{b\}, \{c\}, \{d\}\}$ (算法第7行)
- 生成长度为2的项集并剪枝：$H_2 = \{\{bc\}, \{bd\}, \{cd\}\}$ (算法第10-11行)
- 可以生成3条候选规则：ad→bc，ac→bd，ab→cd
- 假设第3条候选规则的置信度小于阈值，$H_2 = \{\{bc\}, \{bd\}\}$ (算法第7行)
- 生成长度为3的项集：$H_3 = \{\{bcd\}\}$ (算法第10行)
- 对长度为3的项集剪枝：$H_3 = \{\}$ (算法第11行)，算法结束。

目 录

4.关联模式的评估

1. 基本概念

- 虽然先验原理大大减少了候选项集的指数搜索空间，但关联分析算法仍然具备产生大量模式的潜力。
- 由于真正的商业数据库的数据量和维数都非常大，很容易产生数以千计甚至是数以百万计的模式，而其中很大一部分可能是人们不感兴趣的。
- 从众多可能的模式中识别出最有趣的一个并非一项平凡的任务，因为“一个人的垃圾可能是另一个人的财富”。
- 建立一组广泛接受的评价关联模式质量的标准是非常重要的。

4.关联模式的评估

1. 基本概念

- 第一套标准可以通过数据驱动的方法来定义客观兴趣度度量(objective interestingness measure)。
- 这些度量可以用来对模式-项集或规则进行排序，从而为处理从数据集中发现的大量模式提供了一种直接的方法。
- 客观兴趣度度量的例子包括支持度、置信度和相关性。

4.关联模式的评估

1. 基本概念

- 第二组标准可以通过主观论据建立，即模式被主观地认为是无趣的，除非它能够揭示意想不到的信息或提供能导致有益行动的有用信息。
- 例如，尽管规则{黄油} → {面包}有很高的支持度和置信度，但它可能不是有趣的，因为它表示的关系显而易见。
- 另一方面，规则{尿布} → {啤酒}是有趣的，因为这种联系十分出乎意料，并且可能为零售商提供新的交叉销售机会。
- 将主观知识加入到模式的评价中是一项困难的任务，因为需要来自领域专家的大量先验信息。

4.关联模式的评估

2. 兴趣度的客观度量

■ 客观度量是一种评估关联模式质量的数据驱动方法。它不依赖于领域，只需要用户设置阈值来过滤低质量的模式。客观度量常常基于列联表(contingency table)中列出的频度计数来计算。

列联表

	B	$\bar{B}$	
A	f_{11}	f_{10}	f_{1+}
$\bar{A}$	f_{01}	f_{00}	f_{0+}
	f_{+1}	f_{+0}	N

f_{11}: A和B同时出现在一个事务中的次数
f_{01}: 包含B但不包含A的事务的个数
f_{1+}：A的支持度计数
f_{+1}：B的支持度计数

4.关联模式的评估

2. 兴趣度的客观度量

- 支持度-置信度框架的局限性：经典的关联规则挖掘算法依赖支持度和置信度来去除没有意义的模式。
- 支持度的缺点在于许多潜在有意义的模式由于包含支持度小的项而被删。
- 一个变量的支持度度量了其发生的概率，而一对变量A和B的支持度$s(A，B)$度量了两个变量一起发生的概率。因此，联合概率$P(A，B)$可写为：

$$P(A,B) = s(A,B) = \frac{f_{11}}{N}$$

4.关联模式的评估

2. 兴趣度的客观度量

■ 如果假设A和B是统计独立的，即发生A和发生B之间不存在任何联系，则$P(A,B) = P(A) \times P(B)$。因此，在$A$和$B$ 是统计独立的假设下，A和B的支持度 $s_{indep}(A，B)$可写为：

$$s_{indep}(A，B) = s(A) \times s(B) = \frac{f_{1+}}{N} \times \frac{f_{+1}}{N}$$

4.关联模式的评估

$$s(A,B)=\frac{f_{11}}{N} \qquad s_{indep}(A，B)=\frac{f_{1+}}{N}\times\frac{f_{+1}}{N}$$

2. 兴趣度的客观度量

- 如果两个变量$s(A，B)$之间的支持等于$s_{indep}(A，B)$，那么可以认为A和B彼此不相关。
- 如果$s(A，B)$与$s_{indep}(A，B)$有很大不同，那么A和B最有可能是相互依赖的。
- 因此，$s(A，B)$与$s(A)\times s(B)$的任何偏差都可以看作A和B之间统计关系的一个指标。
- 由于置信度只考虑$s(A，B)$和$s(A)$之间的偏差，而不是和 $s(A)\times s(B)$之间的偏差，它不能解释后果的支持度(即$s(B)$) 。

4.关联模式的评估

$$s(A,B)=\frac{f_{11}}{N} \qquad s_{indep}(A,\ B)=\frac{f_{1+}}{N}\times\frac{f_{+1}}{N}$$

2. 兴趣度的客观度量 --- 兴趣因子

- 兴趣因子，也叫作提升度(lift)，可以定义如下:

- 兴趣因子度量了模式$s(A,B)$的支持度与在统计独立性假设下计算出的基准支持度 $s_{indep}(A,\ B)$的比值。

$$I(A,B)\begin{cases}=1 & A和B是独立的\\ >1 & A和B是正相关的\\ <1 & A和B是负相关的\end{cases}$$

4.关联模式的评估

$$I(A,B)=\frac{N\times f_{11}}{f_{1+}\times f_{+1}}$$

$$I(A,B)\begin{cases}=1 & A和B是独立的\\ >1 & A和B是正相关的\\ <1 & A和B是负相关的\end{cases}$$

2. 兴趣度的客观度量 --- 兴趣因子

	$Coffee$	$\overline{Coffee}$	
Tea	150	50	200
$\overline{Tea}$	650	150	800
	800	200	1000

关联规则: Tea → Coffee

Confidence= f_{11}/f_{1+} =150/200= 0.75

$I(A,B)=\frac{1000*150}{200*800}=0.9375$ 稍微负相关

	$Honey$	$\overline{Honey}$	
Tea	100	100	200
$\overline{Tea}$	20	780	800
	120	880	1000

关联规则: Tea → Honey

Confidence= f_{11}/f_{1+} =100/200= 0.5

$I(A,B)=\frac{1000*100}{200*120}=4.1667$ 较强正相关

兴趣因子比置信度度量具有更多统计学优势，使其成为分析变量之间统计独立性的合适度量方法。

4.关联模式的评估

$$corr(x,y)=\frac{covariance(x,\ y)}{standard\_deviation(x)\times standard\_deviation(y)}=\frac{S_{xy}}{S_xS_y}$$

$$s(A,B)=\frac{f_{11}}{N}\qquad s(A)\times s(B)=\frac{f_{1+}}{N}\times\frac{f_{+1}}{N}$$

2. 兴趣度的客观度量 --- 相关度分析

- 相关度分析是分析一对变量之间关系的最流行方法之一。对于连续变量，相关度用皮尔森相关系数定义。对于二元变量，相关度可以用ϕ系数度量，定义如下：

$$\phi=\frac{f_{11}f_{00}-f_{01}f_{10}}{\sqrt{f_{1+}f_{+1}f_{0+}f_{+0}}}$$

- 根据A、B以及(A，B)的支持度来重写，如下:

$$\phi=\frac{s(A,B)-s(A)\times s(B)}{\sqrt{s(A)\times(1-s(A))\times s(B)\times(1-s(B))}}$$

$$\phi=\begin{cases}0 & \text{没有关联}\\ +1 & \text{完全正相关}\\ -1 & \text{完全负相关}\end{cases}$$

- ϕ系数的值范围为−1~+1。从统计角度来看，相关性捕获了$s(A,B)$和$s_{indep}(A，B)$之间的归一化差异。

4.关联模式的评估

$$\phi = \frac{s(A,B) - s(A) \times s(B)}{\sqrt{s(A) \times (1-s(A)) \times s(B) \times (1-s(B))}}$$

$$\phi = \begin{cases} 0 & \text{没有关联} \\ +1 & \text{完全正相关} \\ -1 & \text{完全负相关} \end{cases}$$

2. 兴趣度的客观度量 --- 相关度分析

$$s(A,B) = \frac{f_{11}}{N} \qquad s(A) \times s(B) = \frac{f_{1+}}{N} \times \frac{f_{+1}}{N}$$

	$Coffee$	$\overline{Coffee}$	
Tea	150	50	200
$\overline{Tea}$	650	150	800
	800	200	1000

关联规则: Tea → Coffee

$$\phi = \frac{s(A,B) - s(A) \times s(B)}{\sqrt{s(A) \times (1-s(A)) \times s(B) \times (1-s(B))}} =$$

$$\frac{0.15 - 0.2 \times 0.8}{\sqrt{0.2 \times (1-0.2) \times 0.8 \times (1-0.8)}} =$$ -0.0625 稍微负相关

	$Honey$	$\overline{Honey}$	
Tea	100	100	200
$\overline{Tea}$	20	780	800
	120	880	1000

关联规则: Tea → Honey

$$\phi = \frac{s(A,B) - s(A) \times s(B)}{\sqrt{s(A) \times (1-s(A)) \times s(B) \times (1-s(B))}} =$$

$$\frac{0.1 - 0.2 \times 0.12}{\sqrt{0.2 \times (1-0.2) \times 0.12 \times (1-0.12)}} =$$ 0.5847 正相关

相关度度量具有统计意义，因此被广泛用于评估变量之间统计独立性的强度。