第6讲
关联规则

邓志鸿
北京大学信息科学技术学院

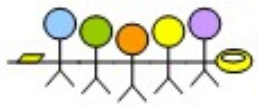
2010年4月12日星期一

火龙果•整理
uml.org.cn

项目2—聚类

- 分类用的测试集
  - 1000条记录
  - 57个属性
- 聚成10个类
  - 按测试数据在测试集中的顺序，给出聚类标示（a,b,c,d,e,f,g,h,i,j等十个字母中的一个，每个字母代表一个聚类），
  - 每个结果占一行。所有聚类结果放在一个文件中。
- 希望各小组及小组成员保持不变

Attention: 分类器_SCY小组准确率为92



项目2—结果样例

测试集		预测结果
0,1.47,0,0, …, 0,2.333,12,21	⟷	b
1.6,0,0,1.3, …, 3.333,13,0,17	⟷	c
0,1,0,1.3, … , 3,13,0,16	⟷	a
1,1,0,1,… , 1,2,3,4	⟷	c
1, 0,0, 1.37, …, 0,2.5,0,13	⟷	e

本项目采用熵作为评价标准

火龙果•整理
uml.org.cn

项目2—Deadline

- 2010年4月20日之前（含20日）提交
  - 源代码：标明组名
  - 报告（PPT）：文件名为组名
    - PPT中注明姓名和学号
    - 同时注明项目分工，按贡献从大到小排序
    - 每组都要在课堂上作报告
  - 聚类结果：文件名为组名
    - **4月27日**将在课堂上隆重发布各组的聚类性能。同时，各小组需要在本次课上做报告。
- 通过email，将上述材料打包发送到（同时抄送给我）
  - gaofuliang@gmail.com
    - 高福亮（电话：13488694361；地址：5410宿舍）

火龙果•整理
uml.org.cn

内容

- 简介
- 基本概念
- 关联分析基本方法
  - 基本内容
  - 频繁模式挖掘
  - 关联规则生成
- 多层关联规则
- 模式评估

火龙果•整理
uml.org.cn

关联规则

- 1993年SIGMOD大会上Agrawal等人首次提出
  - 关联规则挖掘 (association rule mining)
- 目的：发现数据中内在的规律性
  - 人们通常会同时购买什么样的商品？— Beer and diapers?
  - 购买微机后，接下来用户通常会有什么购物行为？
  - 哪种DNA对某个新药敏感？
- 应用
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.
- 核心任务
  - **频繁模式 (Frequent pattern)**
    - 在数据集（库）中频繁出现的模式（项集 (a set of items)、结构 (substructures) 等）。

火龙果•整理
uml.org.cn

内容

- 简介
- 基本概念
- 频繁模式挖掘基本方法
  - 基本内容
  - 频繁模式挖掘
  - 关联规则生成
- 多层关联规则
- 模式评估

火龙果•整理
uml.org.cn

关联规则分析

- Association Rule Analysis
  - 给定事务集合，根据某些项的出现来预测其它项的出现

Market-Basket transactions

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

Example of Association Rules

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

隐含着内在关联，而非偶然现象

火龙果•整理
uml.org.cn

基本概念

- 项 (Item)
  - 最小的处理单位
  - 例如：Bread, Milk
- 事务 (Transaction)
  - 由事务号和项集组成
  - 例如：<1, {Bread,Milk}>
- 事务数据库
  - 由多个事务组成
- 项集 (Itemset)
  - 一个或多个项 (item) 的集
    - 例如：{Milk, Bread, Diaper}
  - k-项集 (k-itemset)
    - 包含k个项的集合

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

火龙果•整理
uml.org.cn

基本概念

- 包含关系
  - 令T为一事务，P为一项集。称T包含P，如果P是T的子集
  - 记T ⊇ P 或 P ⊆ T
- 支持度计数 **(Support count)**
  - 事务数据库中包含某个项集的事务的个数
  - 例如： σ({Milk, Bread,Diaper}) = 2
- 支持度 **(Support)**
  - 事务数据库中包含某个项集的事务占事务总数的比例。
  - 例如： s({Milk, Bread, Diaper}) = 2/5
- 频繁项集 **(Frequent Itemset)**
  - 令P为任何一个项集，称P为频繁项集，如果P的支持度不小于指定的最小阈值 (*minsup* threshold)

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

火龙果•整理
uml.org.cn

基本概念

- 关联规则 **(Association Rule)**
  - 表达形式：X → Y (s, c)
    - 其中， X 和 Y 都是项集，s是规则的支持度，c是置信度
  - 例子：
    {Milk, Diaper} → {Beer} (0.4, 0.67)

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

- 规则评估度量指标
  - 支持度－Support (s)
    - 同时包含X和Y的事务占事务总数的比例
  - 置信度－Confidence (c)
    - 在所有包含X的事务中包含Y的事务所占比例

Example:

$$\{\text{Milk}, \text{Diaper}\} \Rightarrow \{\text{Beer}\}$$

$$s = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{|\text{T}|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk}, \text{Di}}{\sigma(\text{Milk},}$$

火龙果•整理
uml.org.cn

内容

- 简介
- 基本概念
- 关联分析基本方法
  - 基本内容
  - 频繁模式挖掘
  - 关联规则生成
- 多层关联规则
- 模式评估

火龙果•整理
uml.org.cn

关联规则分析—内容

- 给定一个事务数据库TD，关联规则挖掘的目标是要找到所有支持度和置信度都不小于指定阈值的规则。
  - 支持度 ≥ *minsup* threshold
  - 置信度 ≥ *minconf* threshold

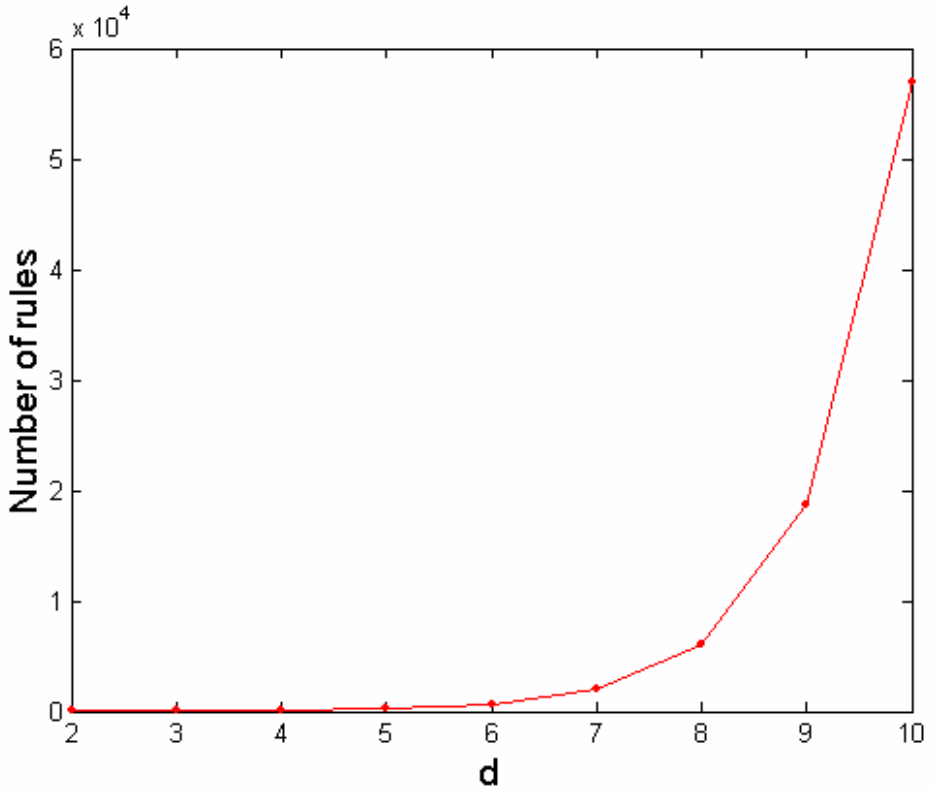
- 穷举法 (Brute-force approach)
  - 列出所有可能的规则
  - 对每条规则计算其支持度和置信度
  - 通过阈值*minsup* 和 *minconf* 过滤无效规则

可计算性？

火龙果•整理
uml.org.cn

计算复杂性分析

- 给定 d 个不同项：
  - 项集数目等于2^d
  - 所有可能的关联规则总数等于：



$$R = \sum_{k=1}^{d-1}\left[\binom{d}{k} \times \sum_{j=1}^{d-k}\binom{d-k}{j}\right]$$

$$= 3^d - 2^{d+1} + 1$$

如果 **d=6** 则 **R = 602**

火龙果•整理
uml.org.cn

关联规则—分析

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

Example of Rules:

{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)

思考

- 所有规则都对应于把同一项集分成两个部分
  {Milk, Diaper, Beer}
- 源自同一项集的规则有相同的支持度，但是置信度不同
- 因此，我们可以分别处理对支持度和置信度的要求
- $X \rightarrow Y$ $s=s(X \cup Y)$ $c= s(X \cup Y) / s(X)$

火龙果•整理
uml.org.cn

关联规则分析

- 分两步执行：
  1. 挖掘频繁项集
     - 生成所有支持度≥ minsup的项集
  2. 构造规则
     - 用每个频繁项集生成高置信度的规则
     - 对频繁模式的每次分割（一分为二）就形成一条规则，再判断该规则是否满足最小置信度阈值条件。

关联规则分析的核心

{Milk, Diaper, Beer} s=0.4

{Milk} s=0.8

{Milk} → {Diaper,Beer}
s=0.4, c=0.4/0.8=0.5

- 但是，挖掘频繁模式仍然是一个“计算昂贵”的工作。

火龙果•整理
uml.org.cn

内容

- 简介
- 基本概念
- **关联分析基本方法**
  - 基本内容
  - **频繁模式挖掘**
  - 关联规则生成
- 多层关联规则
- 模式评估

火龙果•整理
uml.org.cn

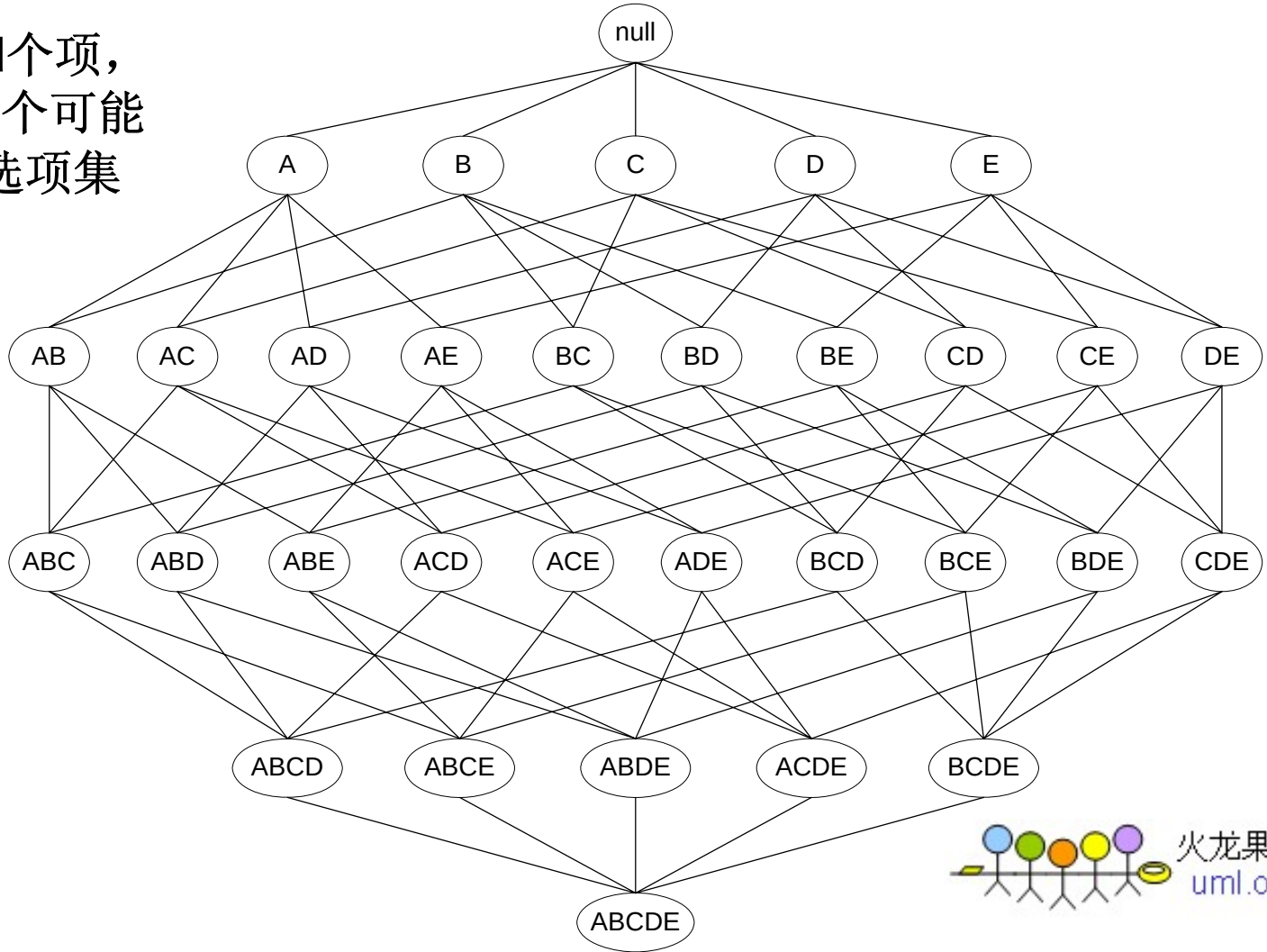
频繁模式挖掘—重要性

- 发现数据集中的有价值的重要性质
- 是其它数据挖掘任务的基础
  - 关联分析：Association rules analysis → Mining Frequent Itemset
  - 因果分析：causality analysis
  - 序列、结构模式：Sequential, structural (e.g., sub-graph) patterns
  - 时空、多媒体、时间序列、流数据模式分析
  - 分类
  - 聚类
  - 数据仓库
  - 语义数据压缩
  - 推荐系统等其他应用

火龙果•整理
uml.org.cn

生成频繁项集

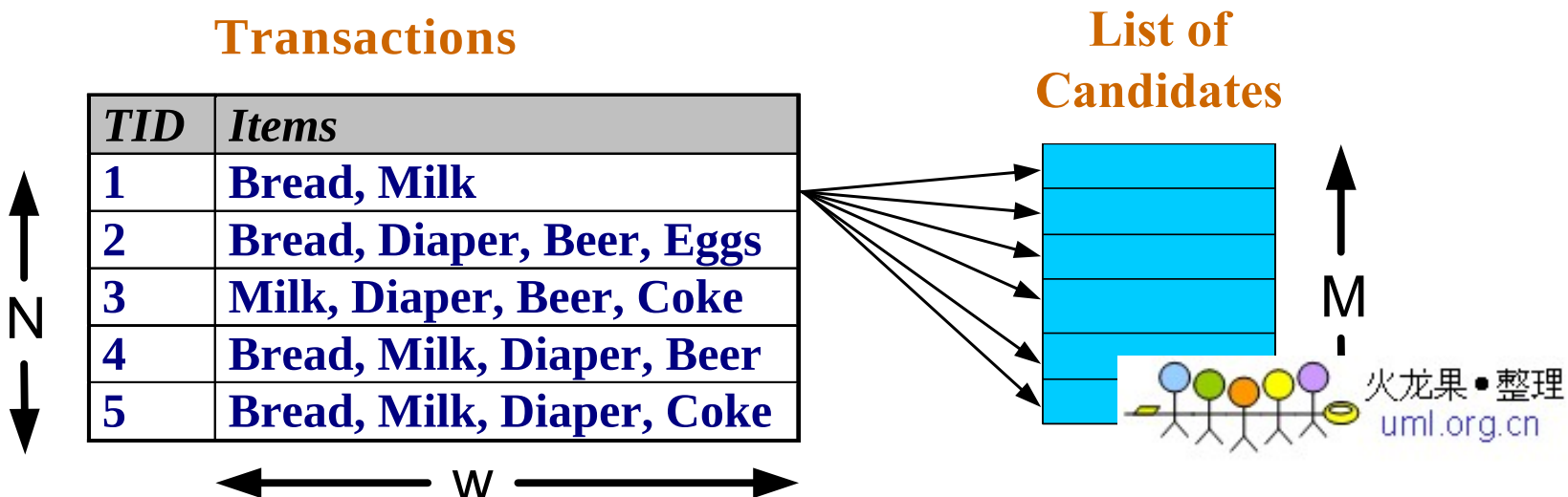
给定**d**个项，有 2^d 个可能的候选项集



火龙果•整理
uml.org.cn

生成频繁项集

- 穷举法 (Brute-force approach)
  - 网格中每个项集都是候选的频繁项集
  - 通过扫描一次数据库，可以得到每个候选项集的支持度
    - 比较每一条事务和每个候选项集
  - 计算复杂度—O(NMw)
    - N为事务数目， $M = 2^d$ 为候选项集， w为一次比较的计算代价



生成频繁项集—策略

- 缩小候选项集的数量 (M)
  - 完全搜索：$M=2^d$
  - 通过裁减技术减少M
- 缩小比较次数 (NM)
  - 用不同的数据结构来存储后续项集和事务
  - 避免比较每一对候选项集和事务
- 缩小比较代价 (w)
  - 采用 DHP 和 vertical-based 挖掘技术

火龙果•整理
uml.org.cn

缩小候选项集

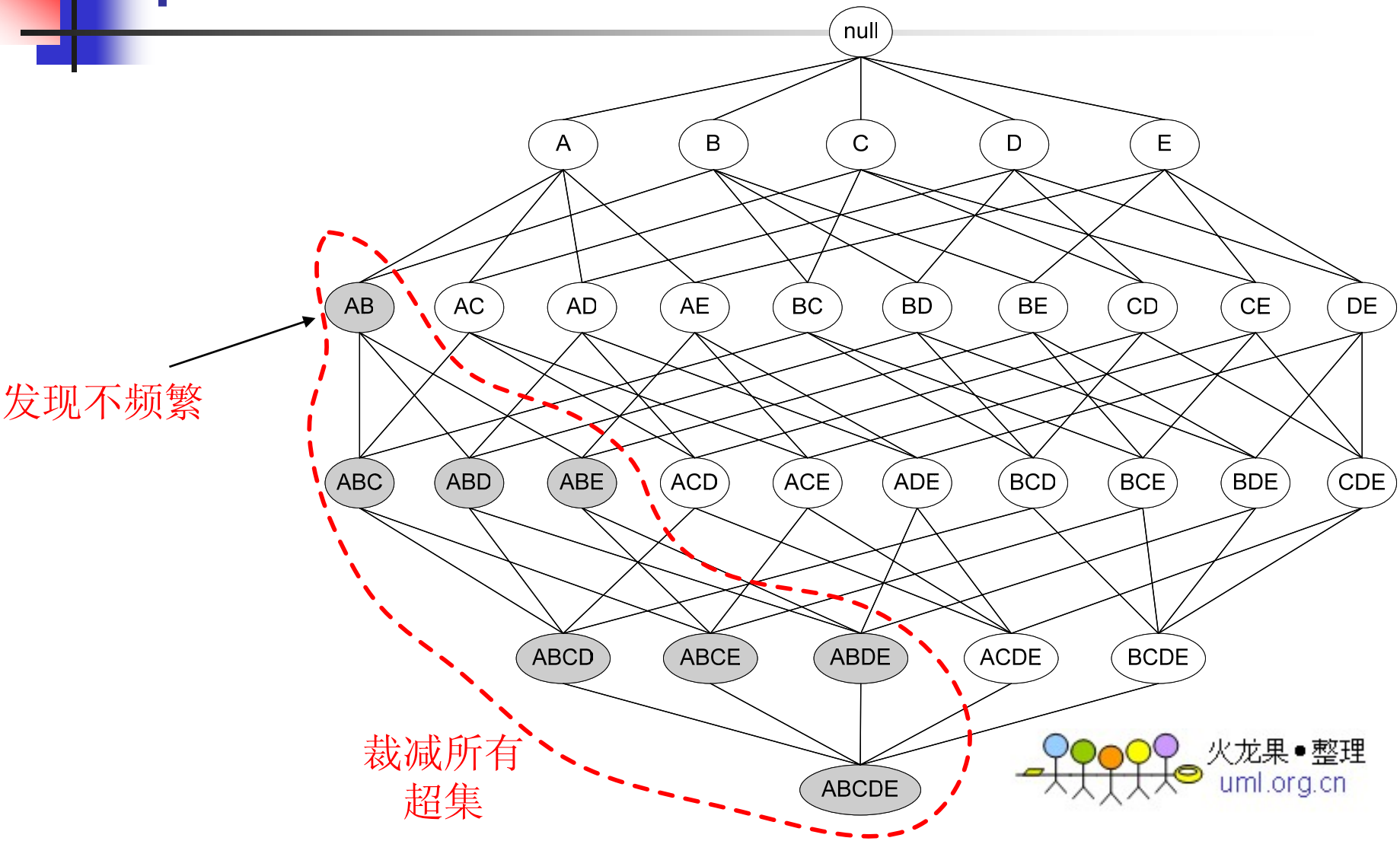
- Apriori 性质:
  - 如果一个项集是频繁的，那么它的所有子集都是频繁的。
  - 也称为反单调性
- Apriori 性质成立的原因如下：

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

  - 任何一个项集的支持度不可能超过其子集的支持度

火龙果•整理
uml.org.cn

Apriori 性质—演示



Apriori 性质—演示

Minimum Support = 3

候选1项集(1-itemsets)

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

候选2项集(2-itemsets)
注意：不需要生成包含Coke或Eggs的2项集

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

候选3项集(3-itemsets)

Itemset	Count
{Bread,Milk,Diaper}	3

如果考虑每个子集，则有：

$C^1_6 + C^2_6 + C^3_6 = 41$

利用Apriori裁剪，只需

6 + 6 + 1 = 13

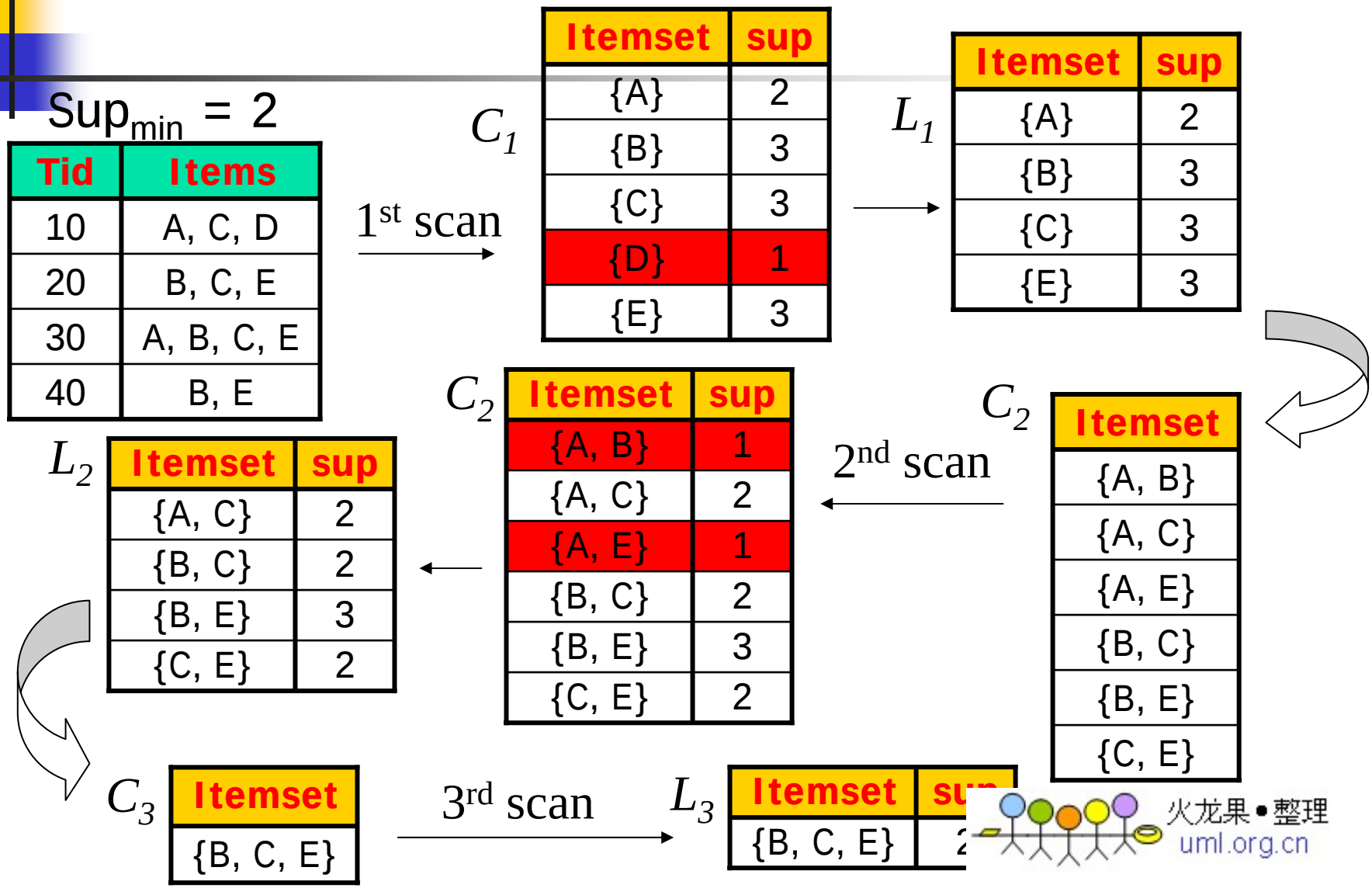
火龙果•整理
uml.org.cn

Apriori 算法—思想

- Method:
  - Let k=1
  - Generate frequent itemsets of length 1
  - Repeat until no new frequent itemsets are identified
    - Generate length (k+1) candidate itemsets from length k frequent itemsets
    - Prune candidate itemsets containing subsets of length k that are infrequent
    - Count the support of each candidate by scanning the DB
    - Eliminate candidates that are infrequent, leaving only those that are frequent

火龙果•整理
uml.org.cn

Apriori 算法—示例



Apriori 算法—（伪码）描述

C_k: Candidate itemset of size k
L_k : frequent itemset of size k

L_1 = {frequent items};
for (k = 1; L_k !=∅; k++) **do begin**
    C_{k+1} = candidates generated from L_k;
    for each transaction t in database do
        increment the count of all candidates in C_{k+1} that are contained in t
    L_{k+1} = candidates in C_{k+1} with support ≥ min_support
    end
return $\cup_k L_k$;

火龙果•整理
uml.org.cn

Apriori 算法—重要细节

- 如何生成候选项集？
  - Step 1: 自连接 (self-joining) L_k
  - Step 2: 裁减 (pruning)
- 如何计算候选项集的支持度？
- 例子
  - L_3={*abc, abd, acd, ace, bcd*}
  - Self-joining: L_3*L_3
    - 由*abc* 和 *abd* 生成*abcd*
    - 由*acd* 和 *ace* 生成*acde*
  - Pruning:
    - 因为*ade*不在L_3中，所以不用处理*acde* （它不可能是频繁项集）
  - C_4={*abcd*}

火龙果•整理
uml.org.cn

Apriori 算法—生成候选项集

- Suppose the items in L_{k-1} are listed in an order
- Step 1: self-joining L_{k-1}

  insert into C_k

  select $p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}$

  from $L_{k-1}\ p, L_{k-1}\ q$

  where $p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$
- Step 2: pruning

  For all ***itemsets c in*** C_k do

  For all ***(k-1)-subsets s of c*** do

  if *(s is not in* L_{k-1}*)* **then delete** *c*

火龙果•整理
uml.org.cn

Apriori 算法—计算候选项集支持度

- 为什么计算候选项集会是一个难题？
  - 候选项集的数目仍然可能很大
  - 一个事务中可能包含多个候选项集
- 思路
  - 减少比较次数
- 方法：
  - 用hash树来存储候选项集
  - hash树的叶子结点包含候选项集及其计数
  - 非叶结点包含hash table
  - hash函数： 找出包含在一个事务中的

火龙果•整理
uml.org.cn

构造Hash树

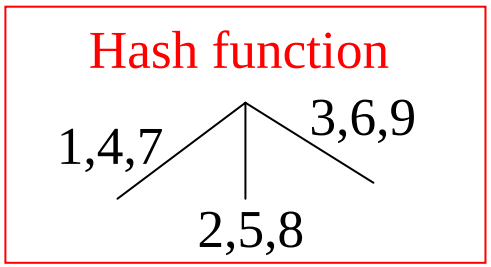
假定有**15**个长度为**3**的候选项集：

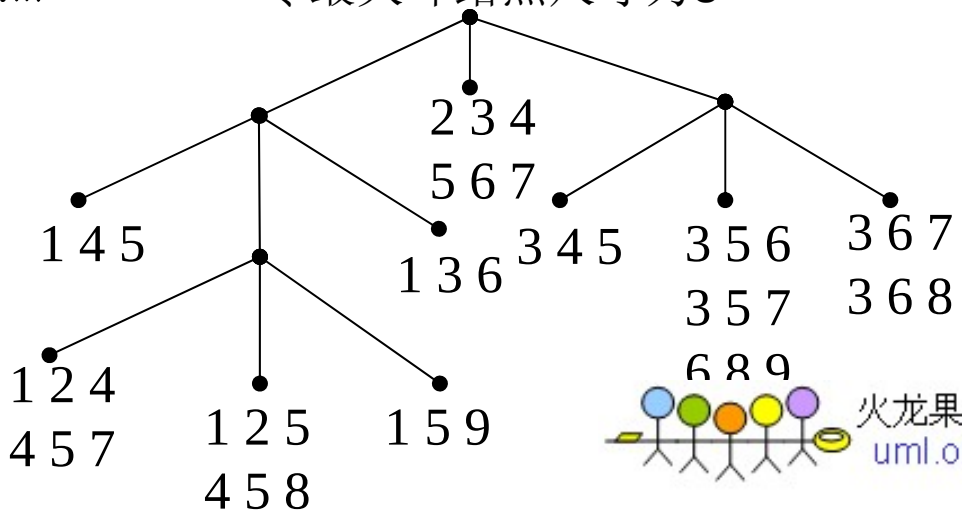
{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}

设定：

- **Hash**函数
- 最大叶结点尺寸：在一个叶结点中存储的最大候选项集的数目（如果大于该阈值则分裂该结点）

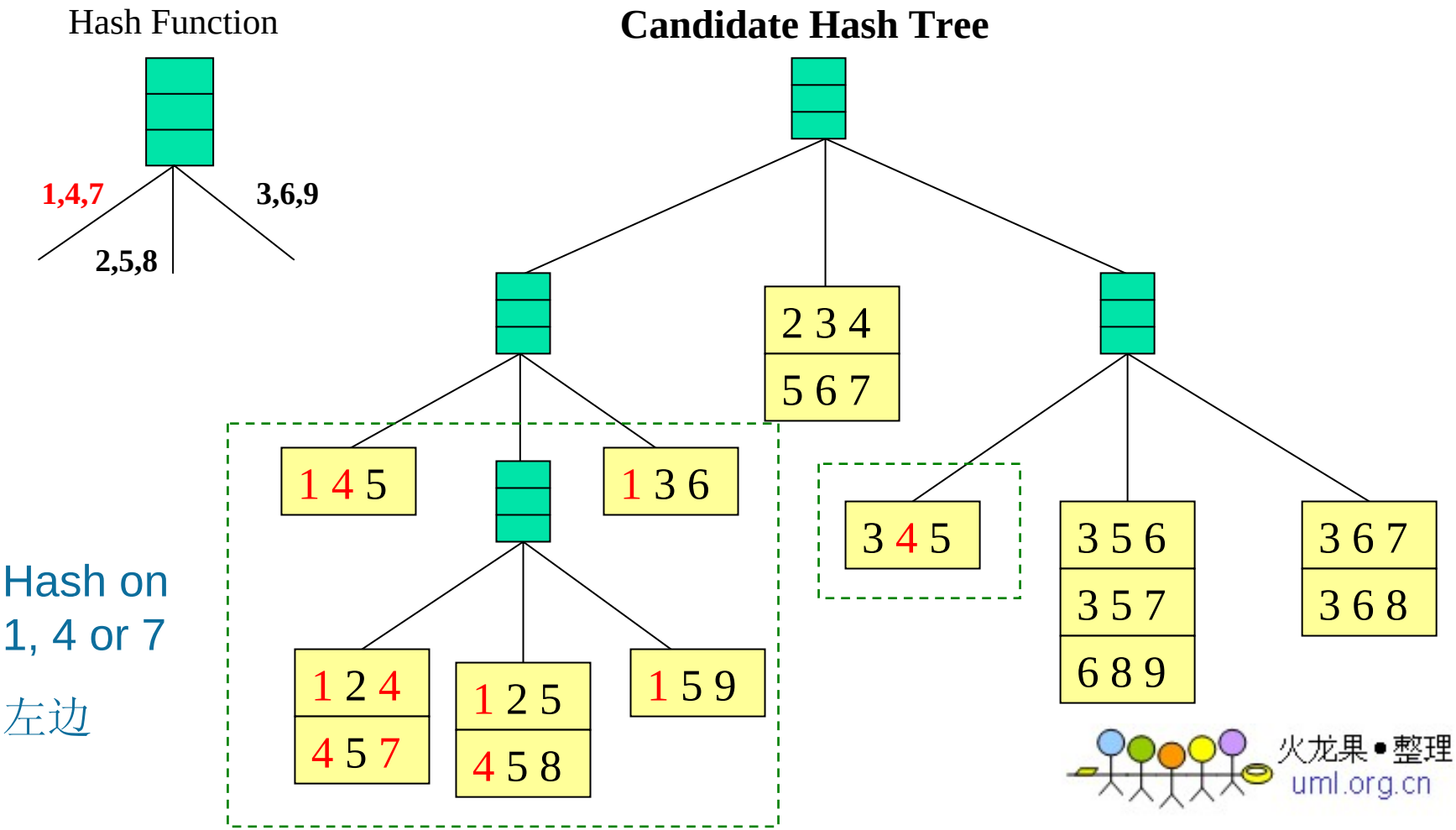
令最大叶结点尺寸为3





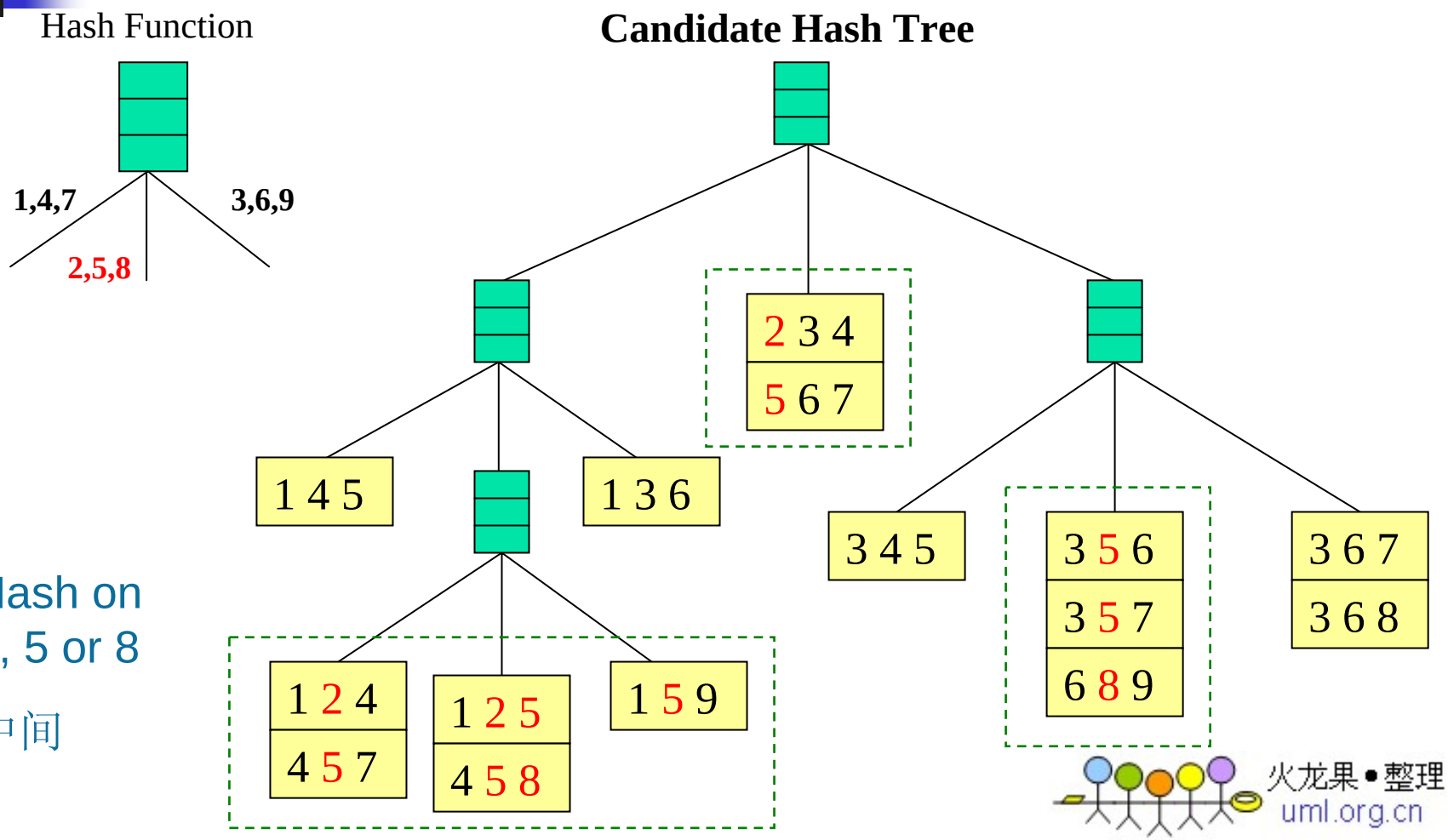
火龙果•整理
uml.org.cn

Hash树



火龙果•整理
uml.org.cn

Hash树

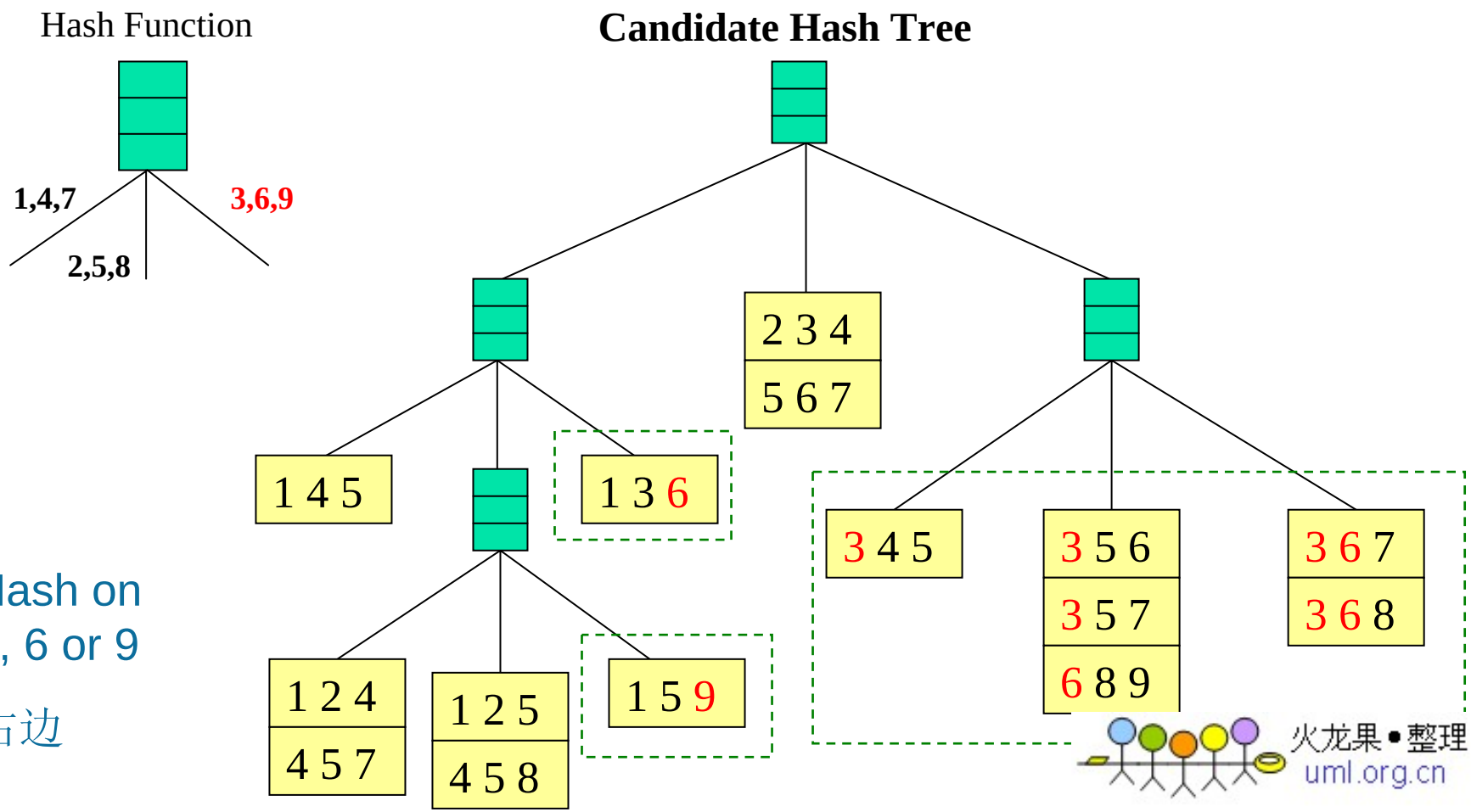


Hash on
2, 5 or 8

中间

火龙果•整理
uml.org.cn

Hash树



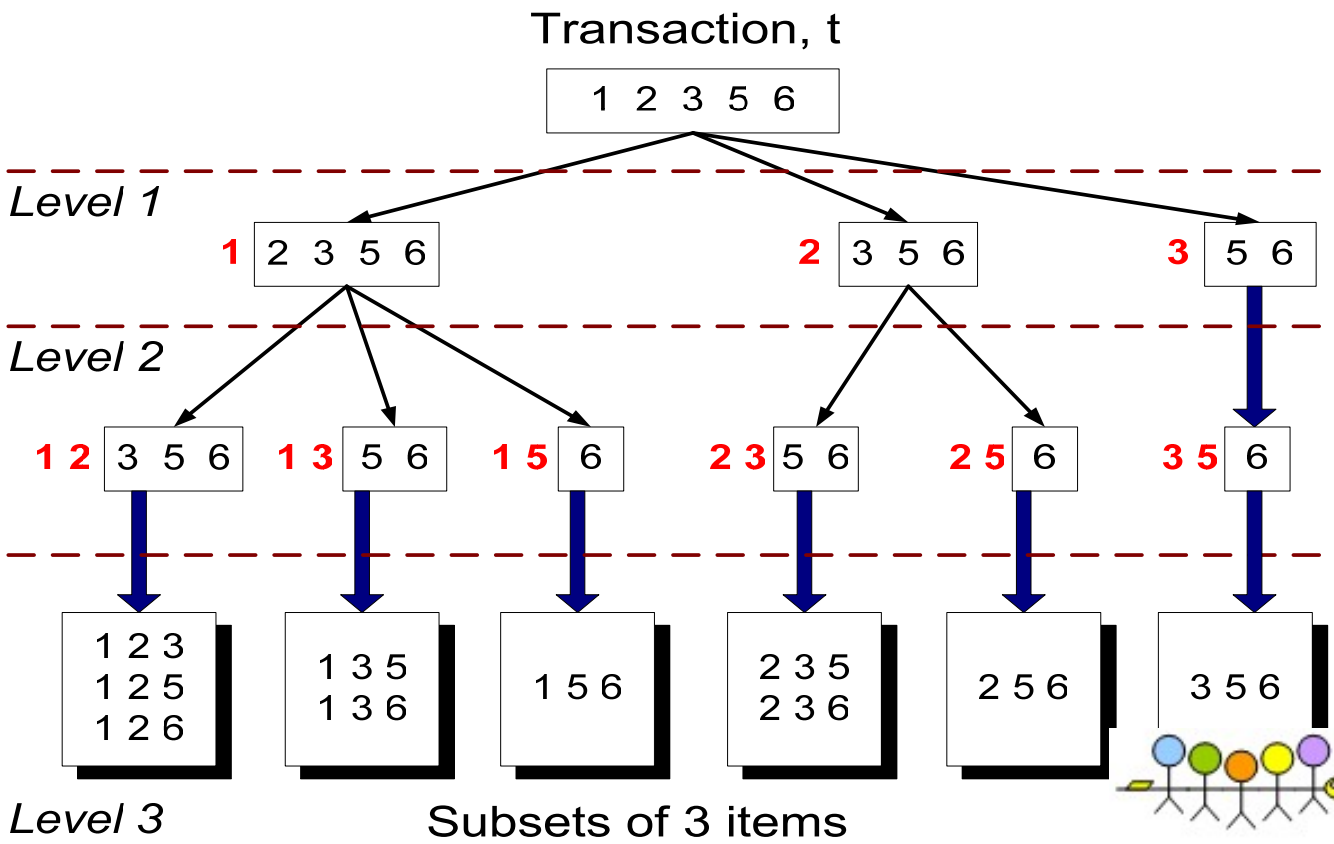
Hash on
3, 6 or 9

右边

火龙果•整理
uml.org.cn

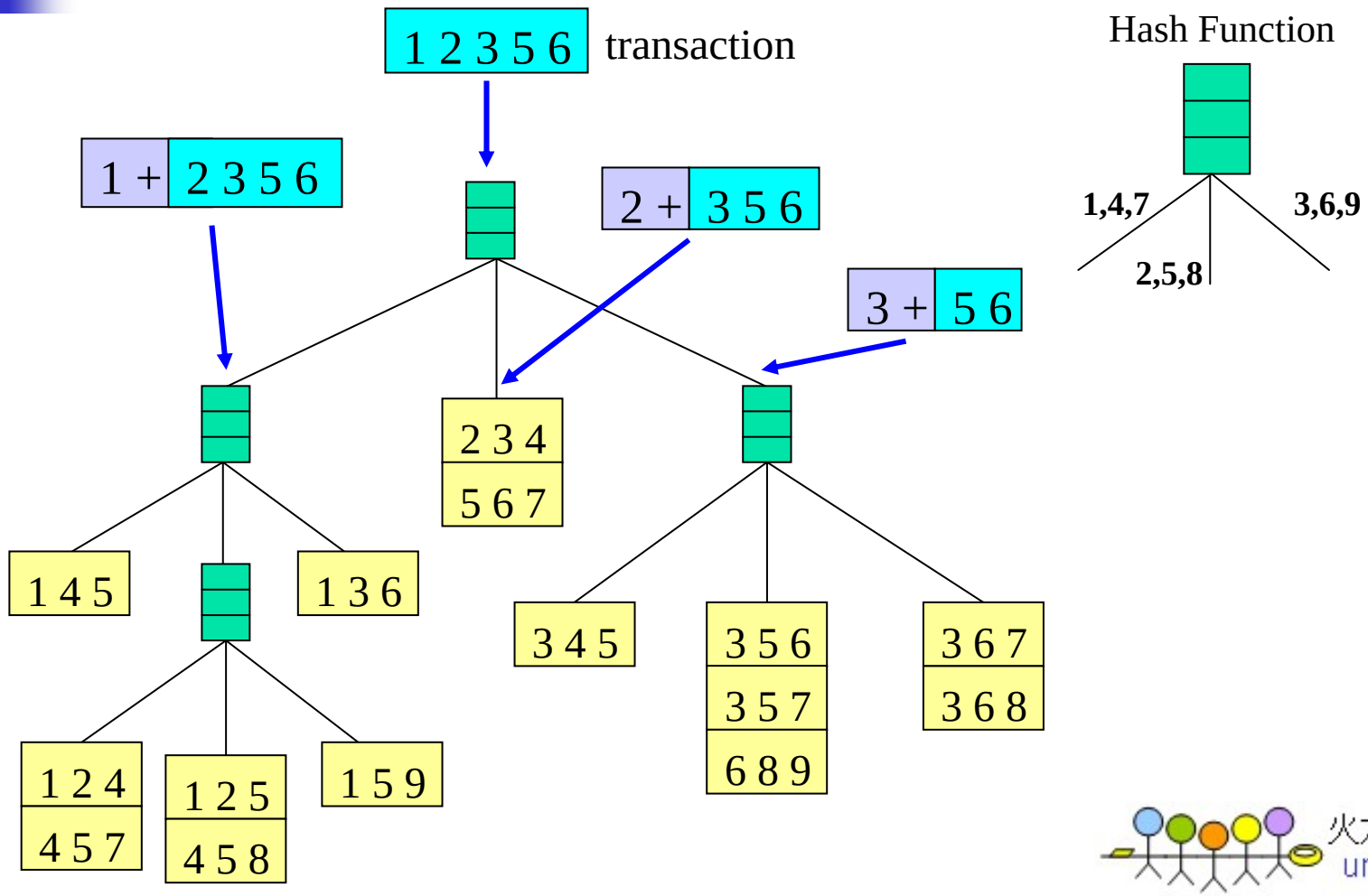
子集操作

给定一个事务，其可能的
长度为3的子集有哪些？



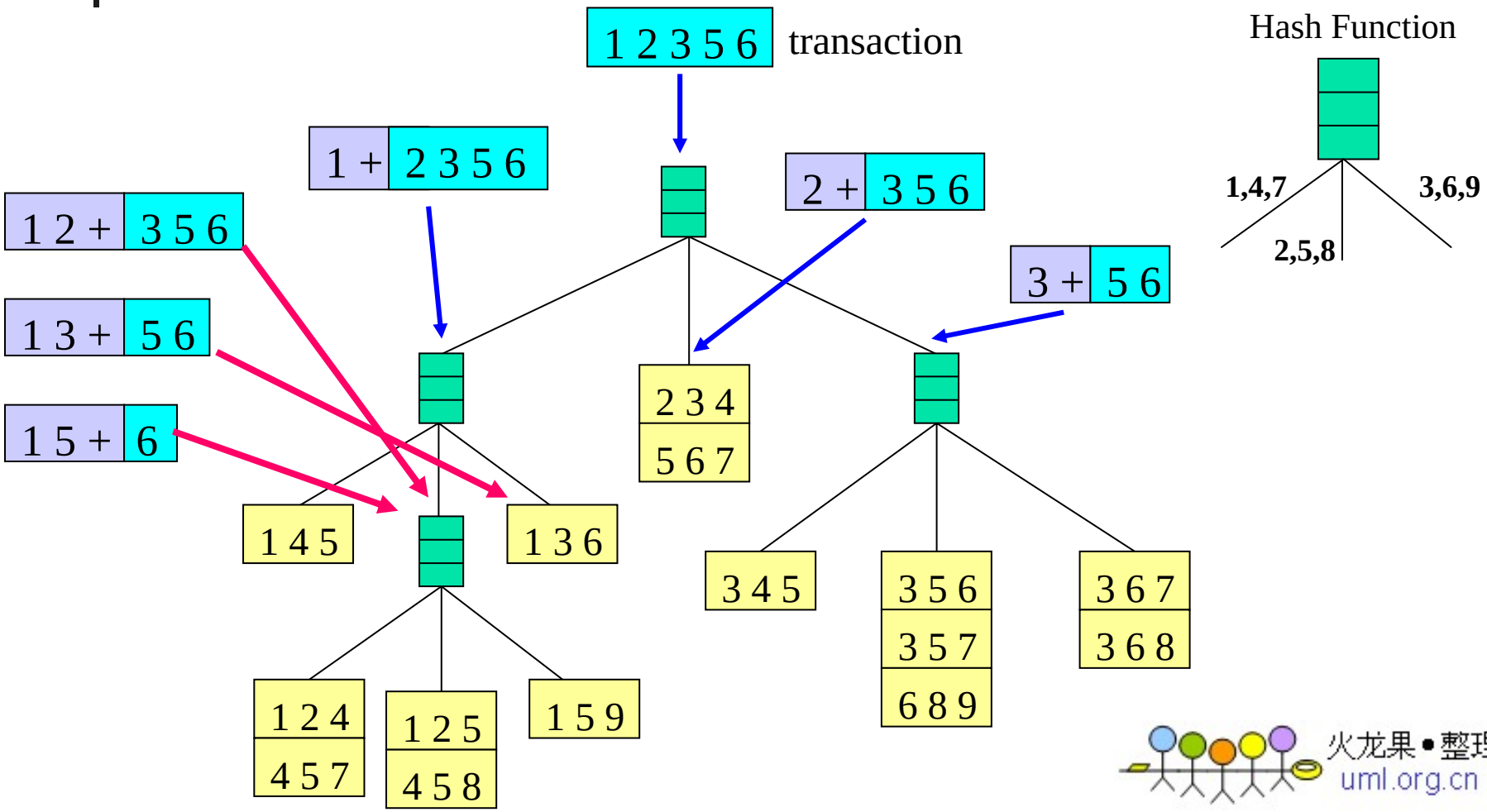
火龙果•整理
uml.org.cn

Hash树—子集操作



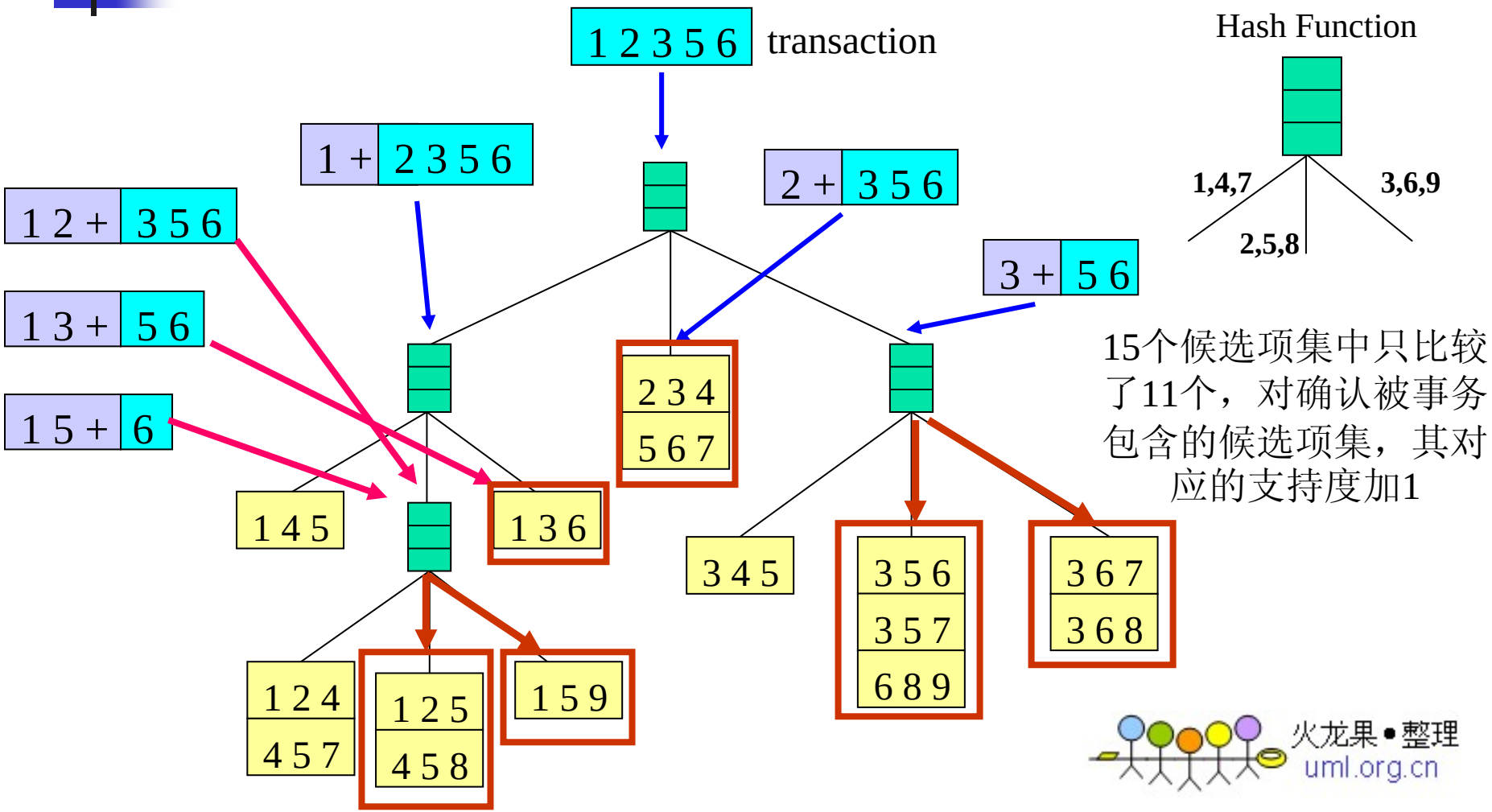
火龙果•整理
uml.org.cn

Hash树—子集操作



火龙果•整理
uml.org.cn

Hash树—子集操作



15个候选项集中只比较了11个，对确认被事务包含的候选项集，其对应的支持度加1

火龙果•整理
uml.org.cn

挖掘频繁2-项集

- 很多实验表明，在Apriori 算法（或基于Apriori的算法）频繁模式挖掘过程中，频繁2-项集的查找最费时间
  - 频繁1-项集很多，所以候选2-项集就非常多

$$C_n^2$$

  - 方法
    - 在扫描事务数据库时，同时计算每个事务所包含2-项集的支持度。
    - 第一次扫描数据库，同时找到频繁1-项集和频繁2-项集

火龙果•整理
uml.org.cn

发现频繁2-项集—示例

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

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

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

A, C, D → {A, C}, {A, D}, {C, D}

B, C, E → {B, C}, {B, E}, {C, E}

...

火龙果•整理
uml.org.cn

Apriori算法－挑战

- 存在问题
  - 大量的候选项集
  - 候选支持度计算代价高
- 改进
  - 减少扫描的数据量
  - 减少候选项集数目

火龙果•整理
uml.org.cn

减少扫描的数据量

- Apriori算法每次循环都要扫描一遍数据库，用来计算候选项集的支持度。随着项集长度增加，候选项集的个数逐渐减少，包含这些候选项集的事务也越来越少，但是扫描的事务量并没有改变。
- 提高效率的方法：在后续循环中逐渐减少扫描的事务。
  - AprioriTid
  - Eclat

火龙果•整理
uml.org.cn

AprioriTid 算法—理论基础

- 基本定理
  - 如果一个事务不包含频繁k-项集，那么该事务必然不包含频繁(k+1)-项集。
- 由以上定理可知，把不包含频繁k-项集的事务删除后，不会影响计算长度更长（>k）的项集的支持度。
- 基于上述思想构成了AprioriTid 算法

证明？

火龙果•整理
uml.org.cn

AprioriTid —基本思想

- 在产生候选项集之后，构造一个Tid表，用来记录每个事务包含的候选项集。候选k-项集的Tid表记做$\underline{C}^k$，其形式为<t.Tid, $\{C \in C_k \mid C \subseteq t\}$>，其中， Tid是事务t的标识， C是事务t中包含的候选k-项集。如果一个事务不包含任何候选k-项集，则这个事务就不会出现在$\underline{C}^k$中。
- 对于k＝1， $\underline{C}^1$ ＝TD；
- 对于k > 1， $\underline{C}^k$ 由$\underline{C}^{k-1}$生成
  - 如果一个事务包含了一个候选k-项集的两个频繁(k－1)-项集，那么其必然包含这个候选k-项集。
  - 故构造$\underline{C}^k$的方法如下：
    - 对每个候选k-项集P，如果P的两个频繁(k－1)-项集都包含在$\underline{C}^{k-1}$里的某个记录中，则添加P到$\underline{C}^k$的相应记录中。

火龙果•整理
uml.org.cn

AprioriTid 算法—算法描述

- Pseudo-code:

  C_k: Candidate itemset of size k; L_k : frequent itemset of size k

  L_1 = {frequent items};

  $\underline{C}^1$ = TD;

  for (k = 1; L_k !=∅; k++) **do begin**

  C_{k+1} = candidates generated from L_k;

  $\underline{C}^{k+1}$ = ∅;

  for each t in $\underline{C}^k$ do

  C_t ={$C \in C_{k+1}$ | C[1] C[2]... C[k] $\in t$ and C[1] C[2]... C[k-1] C[k+1] $\in t$ }

  for all $C \in C_t$,increment the count of C by 1;

  if C_t !=∅ then $\underline{C}^{k+1}$ = $\underline{C}^{k+1} \cup$ {<t.Tid, C_t >}

  end for

  L_{k+1} = candidates in C_{k+1} with min_support

  end for

  return $\cup_k L_k$;

火龙果•整理
uml.org.cn

AprioriTid算法—示例

Minsup =3

C^1

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

频繁1-项集

项集	支持度
{B}	4
{C}	4
{E}	3

候选2-项集

项集
{BC}
{BE}
{CE}

C^2

Tid	Items
20	{BC}, {BE}, {CE}
30	{BC}, {BE}, {CE}
40	{BE}
50	{BC}

频繁2-项集

项集	支持度
{BC}	3
{BE}	3

火龙果•整理
uml.org.cn

AprioriTid 算法—讨论

- 优点
  - 用逐渐减少的Tid表代替原来的事务数据库
- 缺点
  - 在初始阶段，尤其是第二次循环（发现频繁2-项集），候选项集的个数非常多，导致构造的Tid表可能比原事务数据库还要大很多。这时Apriori在效率上要优于AprioriTid
- AprioriHybral
  - 结合Apriori和AprioriTid的优点
  - 思想
    - 首先采用Apriori算法，同时估计Tid表的大小。
    - 当Tid表减小到可以载入内存时，就转而采用AprioriTid算法

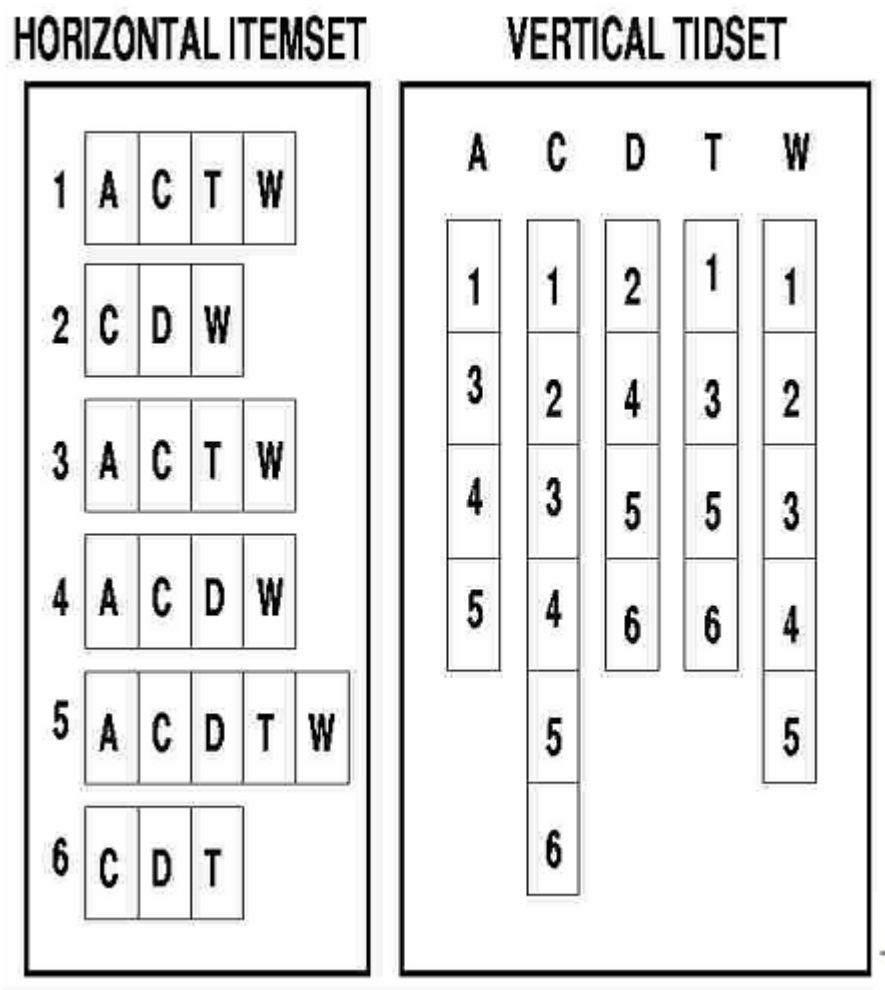
火龙果•整理
uml.org.cn

Eclat

- 垂直数据表示 (vertical data format)
  - Tidset
- Tidset
  - 令P为一个项集，其Tidset定义如下：
    - Tidset(P)={t.Tid| P ⊆ t}
- 性质1：P的支持度等于Tidset(P)势（包含元素的个数）
- 性质2：令S_k = {Tidset(P) | P是频繁k-项集}，则对任何C∈C_{k+1} (候选k+1项集)， Tidset(C)可由S_k中的两个元素Tidset (X) 和 Tidset (Y)得到，并且Tidset (C)＝ Tidset (X) ∩ Tidset (Y)
  - X＝ C[1] C[2]... C[k-1] C[k]
  - Y＝ C[1] C[2]... C[k-1] C[k+1]
- Eclat算法思想与Apriori基本一致，只不过在获取候选项集的Tidset直接用两个频繁项集的Tidset的交。而候选项集的支持度就直接等于|Tidset|

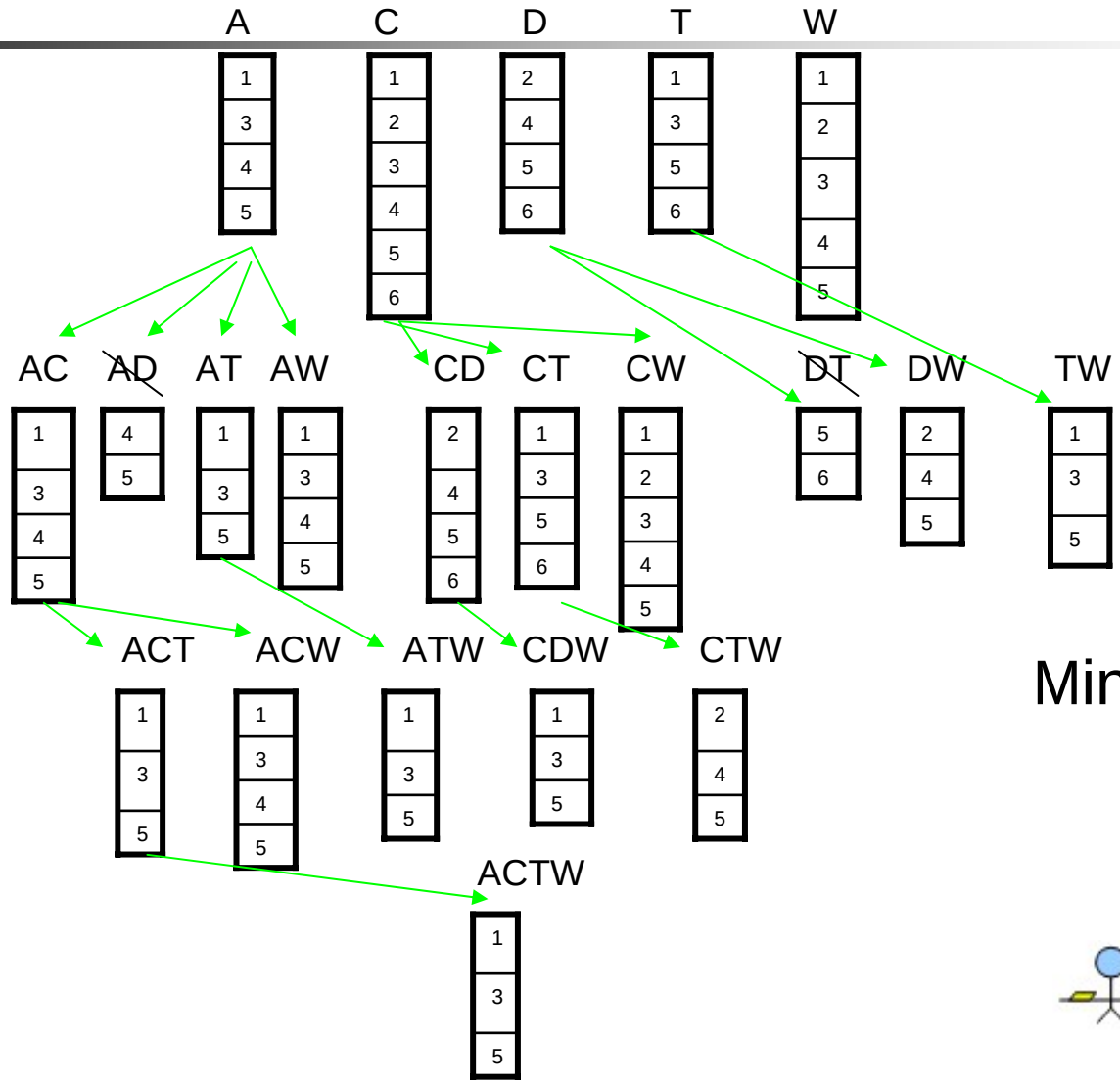
火龙果•整理
uml.org.cn

垂直数据表示



火龙果•整理
uml.org.cn

Eclat算法– 示例



Minsup=3

火龙果•整理
uml.org.cn

减少候选项集数目

- Apriori算法的复杂性与候选项集的数目有关，候选项集越多，构造的hash树越大，运行时间就越长，因此，提高Apriori算法的另外一个途径就是减少候选项集数目。
  - Apriori算法在构造候选k-项集的时候利用了频繁k-1项集进行裁减，有效地降低了候选项集的数量。
  - 但是这个方法对生成候选2-项集基本上没有太大作用。
    - 令频繁1-项集个数为n，则候选2-项集个数为n(n-1)/2
- DHP算法利用hash技术改进了候选项集的生成过程
  - 特别是对候选2-项集
  - Hash表由一系列单元构成，每个单元保存一组项集和一个计数器，用来存储通过hash函数映射到该单元的项集以及总的支持度。函数值相同的项集共享一个单元。显然，每个项集的支持度不会大于它所在单元的计数器的值。

火龙果•整理
uml.org.cn

DHP算法

- 基本思路
  - 在第k次循环时，计算候选k-项集的支持度的同时，搜索每条事务中包含的k+1项集，通过一个hash函数映射到hash表中的某个单元，同时单元的计算器加1。当处理完所有的事务后，hash表也构造完了。
  - 在第k+1次循环时，先由频繁k-项集得到候选k+1项集，然后利用hash表对候选k+1项集进行裁减，方法如下：
    - 如果一个候选项集对应的hash表单元的计数器小于最小阈值，就将其从候选项集中删除。
    - 经过这样的删除，候选项集的数目将显著减少。

火龙果•整理
uml.org.cn

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

On the fly

C_1	count	L_1
{A}	2	{A}
{B}	3	{B}
{C}	3	{C}
{D}	1	{E}
{E}	3	

Making a hash table — minimum support, s = 2

100 {AC},{AD},{CD}

200 {BC},{BE},{CE}

300 {AB},{AC},{AE},{BC},{BE},{CE}

400 {BE}

h({X Y})=((order of x)*10 + (order of y)) mod 7;

0	1	2	3	4	5	6
{CE} {CE} {AD}	{AE}	{BC} {BC}		{BE} {BE}	{AB}	{AC} {CD} {AC}
3	1	2	0	3	1	3

Hash table H_2

Hash address

the number of items hashed to bucket 2

Generating C_2 — # in a bucket with the itemset — C_2

L_1 ? L_1			C_2
	{AB}	1	{AC}
	{AC}	3	{BC}
	{AE}	1	{BE}
	{BC}	2	{CE}
	{BE}	3	
	{CE}	3	

火龙果•整理
uml.org.cn

内容

- 简介
- 基本概念
- **关联分析基本方法**
  - 基本内容
  - 频繁模式挖掘
  - **关联规则生成**
- 多层关联规则
- 模式评估

火龙果•整理
uml.org.cn

生成关联规则

- 给定频繁项集L，找出 L的所有非空子集f， 满足 f → L – f 的置信度不小于最小置信度阈值
  - 如果{A,B,C,D}是频繁项集，则候选的规则有：

ABC →D,	ABD →C,	ACD →B,	BCD →A,
A →BCD,	B →ACD,	C →ABD,	D →ABC
AB →CD,	AC → BD,	AD → BC,	BC →AD,
BD →AC,	CD →AB,		

- 令|L| = k，则有2^k – 2 个候选的关联规则 (不考虑 L → ∅ and ∅ → L)

火龙果•整理
uml.org.cn

生成关联规则

- 如何有效地从频繁项集中生成关联规则？
  - 一般而言，置信度并不具有反单调性质 (anti-monotone property)
  - 例如：c(ABC →D)可能比 c(AB →D) 大，也可能比 c(AB →D) 小

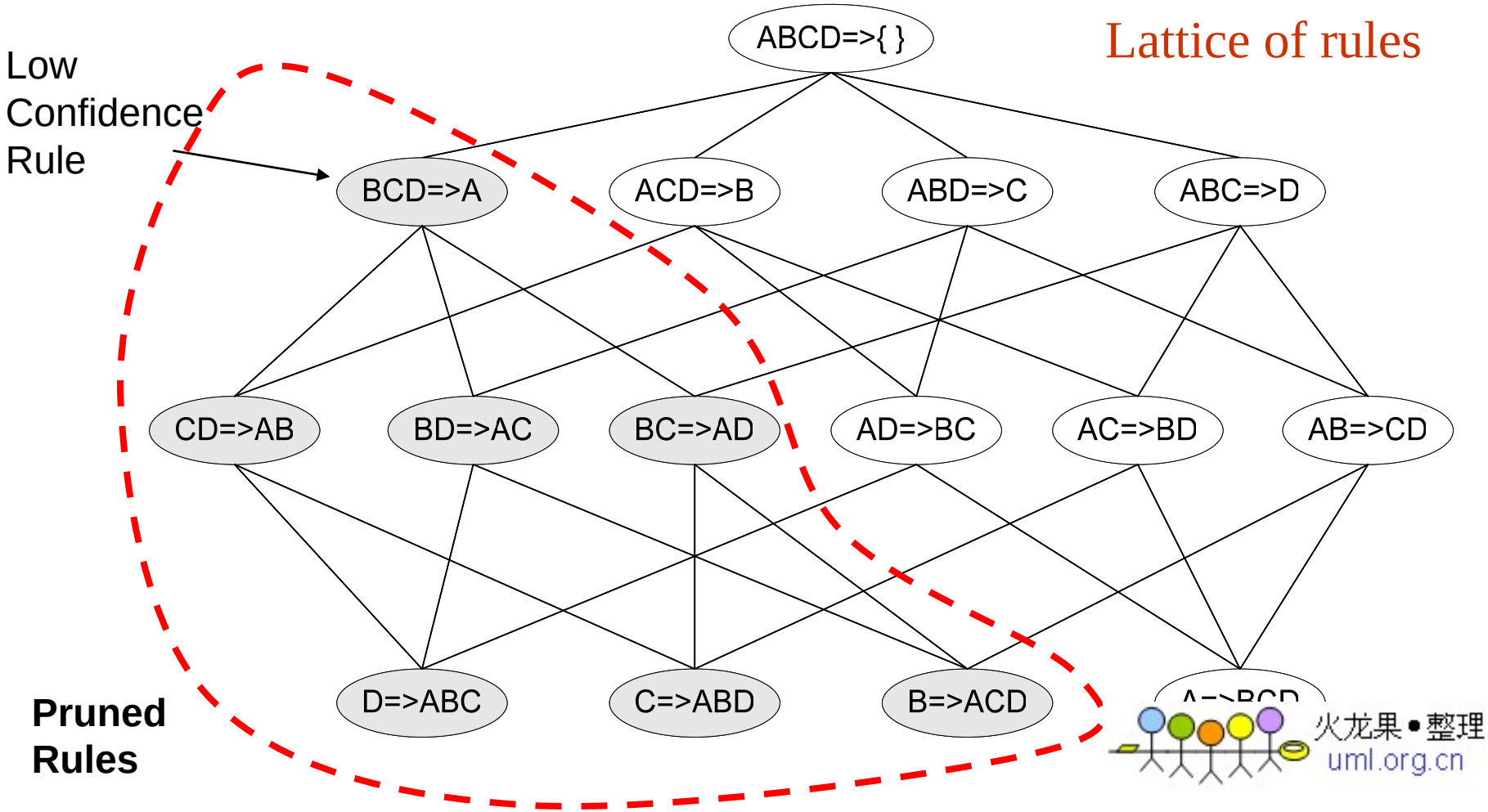
证明

  - 但是，由同一个项集生成的规则（按原顺序）却具有反单调性质
  - 例如：L = {A,B,C,D}:

$$c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$$

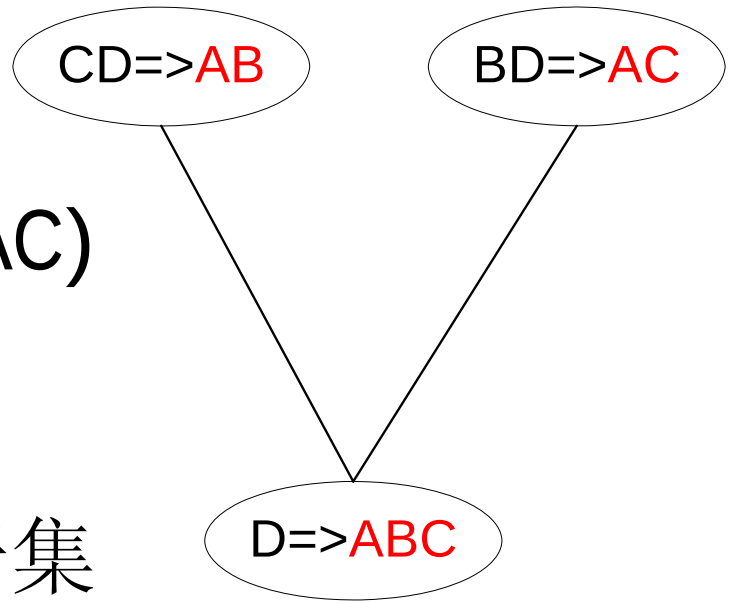
火龙果•整理
uml.org.cn

基于Apriori算法的关联规则生成



基于Apriori算法的关联规则生成

- 通过合并具有共同前缀结论的关联规则生成候选规则



- 合并(CD=>AB,BD=>AC)将生成D => ABC
- 裁减D=>ABC如果其子集AD=>BC置信度小于最小阈值

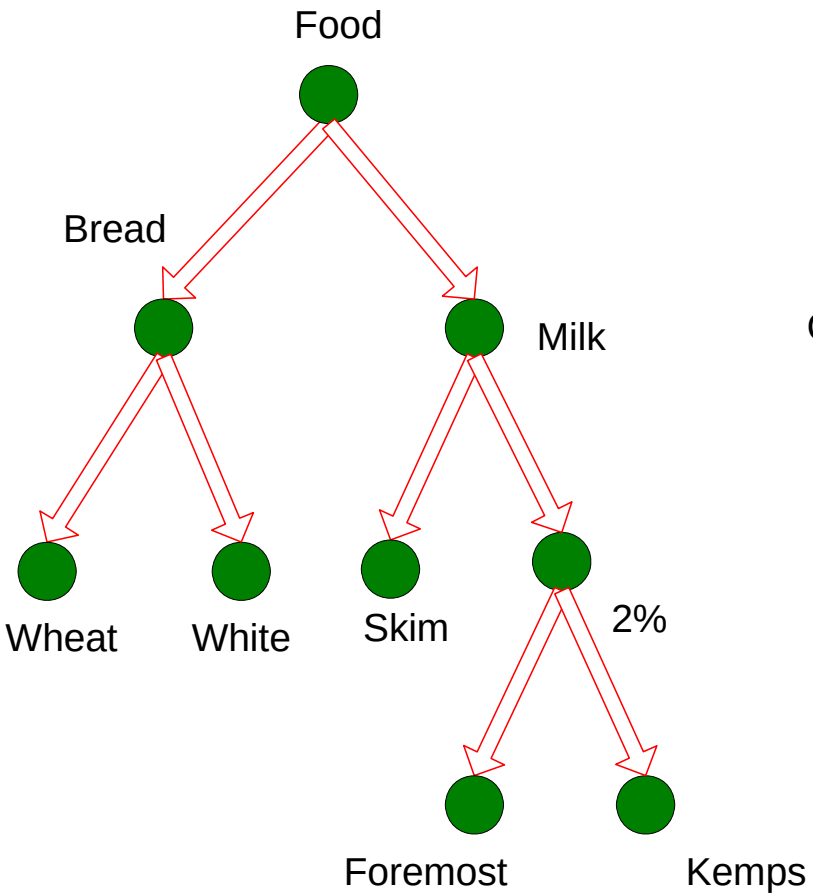
火龙果•整理
uml.org.cn

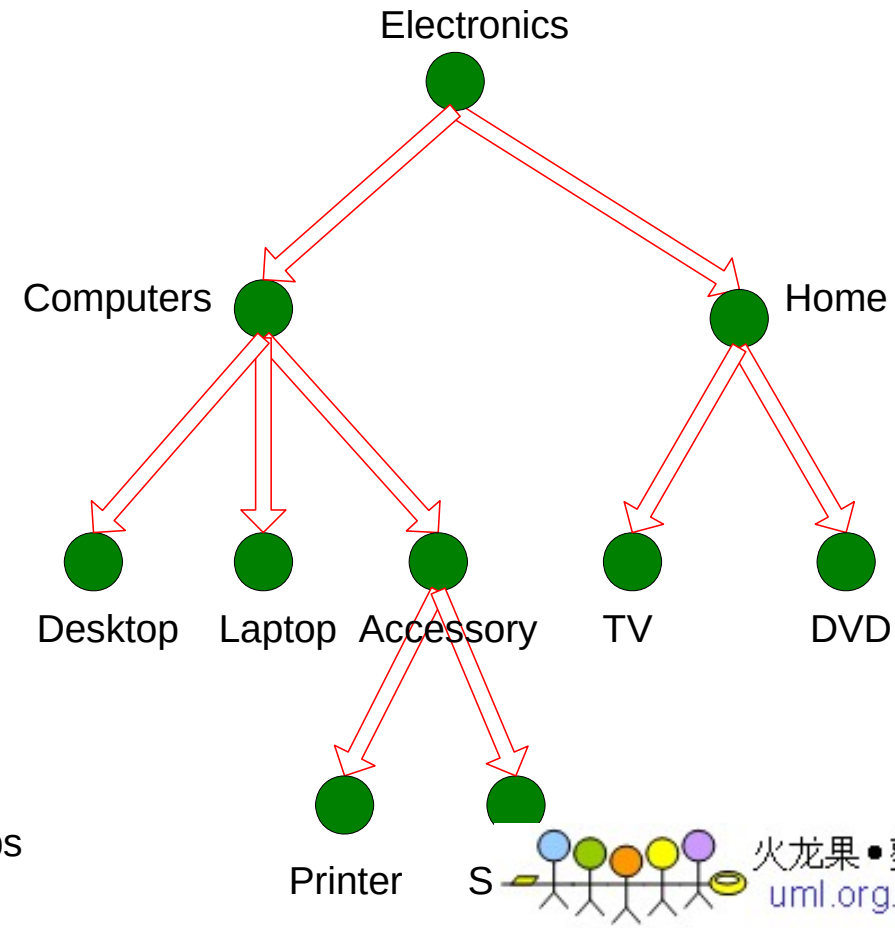
内容

- 简介
- 基本概念
- 关联分析基本方法
  - 基本内容
  - 频繁模式挖掘
  - 关联规则生成
- **多层关联规则**
- 模式评估

火龙果•整理
uml.org.cn

概念层次





火龙果•整理
uml.org.cn

多层关联规则

- 为什么需要考虑概念层次？
  - 底层规则可能由于没有足够的支持度而出现在频繁项集中
  - 概念层次的底层规则太详细了
    - 例如
      - skim milk → white bread
      - milk → wheat bread,
      - skim milk → wheat bread
      - …
    - 其实是暗示 milk 和 bread之间的关联

火龙果•整理
uml.org.cn

多层关联规则

- 遍历概念层次时，支持度和置信度如何变化？
  - 如果X是X1和X2的父概念，那么
    $sup(X) \geqslant sup(X1) + sup(X2)$
  - 如果$sup(X1 \cup Y1) \geqslant minsup$，并且X是X1的父概念， Y是Y1的父概念，则$sup(X \cup Y1) \geqslant minsup$，$sup(X1 \cup Y) \geqslant minsup$， $sup(X \cup Y) \geqslant minsup$
  - 如果$conf(X1 \Rightarrow Y1) \geqslant minconf$，则$conf(X1 \Rightarrow Y) \geqslant minconf$

火龙果•整理
uml.org.cn

多层关联规则

- 方法1
  - 把高层概念（项）添加到每个事务中

  原始事务：
  {skim milk, wheat bread}
  修改后事务:
  {skim milk, wheat bread, milk, bread, food}

- 问题
  - 高层项具有更高的支持度
    - 如果支持度设置过低，则有太多的频繁项集包含高层项
  - 增加了数据的维度

火龙果•整理
uml.org.cn

多层关联规则

- 方法2
  - 首先在最高层生成频繁项集
    - 事务中项均为最高层项
  - 然后，在下一层生成频繁项集，如此直到最底层

- 问题
  - I/O将大量增加
    - 需要更多次扫描数据库
  - 可能遗漏有意义的 cross-level关联模式

火龙果•整理
uml.org.cn

内容

- 简介
- 基本概念
- 关联分析基本方法
  - 基本内容
  - 频繁模式挖掘
  - 关联规则生成
- 多层关联规则
- **模式评估**

火龙果•整理
uml.org.cn

模式评估 (Pattern Evaluation)

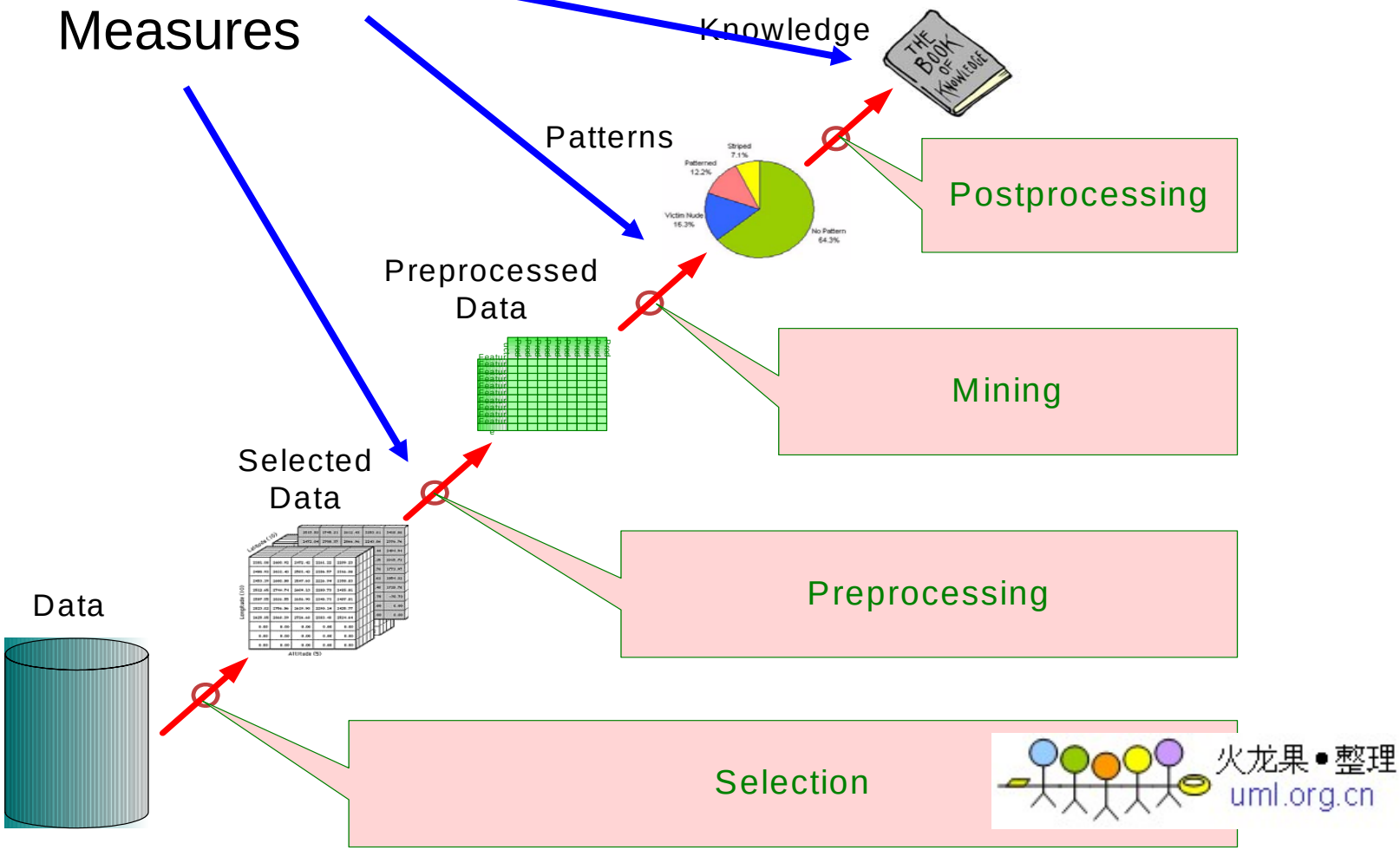
- 关联规则算法能产生大量的规则
  - 其中很多是无意义或是冗余的
  - 冗余
    - 例如：{A,B,C} → {D} 和 {A,B} → {D}有同样的支持度和置信度

- 兴趣度 (Interestingness)
  - 反映模式的重要程度，用于裁减模式或对模式排序
  - 支持度和置信度是用到的两个度量

火龙果•整理
uml.org.cn

兴趣度的应用

Interestingness Measures



火龙果•整理
uml.org.cn

计算兴趣度

- 给定规则 $X \rightarrow Y$，计算规则兴趣度的信息可完全由列联表给出 (contingency table)

$X \rightarrow Y$的列联表

	Y	$\overline{Y}$	
X	f_{11}	f_{10}	f_{1+}
$\overline{X}$	f_{01}	f_{00}	f_{o+}
	f_{+1}	f_{+0}	|T|

f_{11}: X and Y的支持度
f_{10}: $\underline{X}$ and $\overline{Y}$的支持度
f_{01}: $\underline{X}$ and $\underline{Y}$的支持度
f_{00}: X and $\overline{Y}$的支持度

用于定义不同的度量

- support, confid... lift, Gini, J-measure, etc.

火龙果•整理
uml.org.cn

置信度—不足

	Coffee	$\overline{\text{Coffee}}$	
Tea	15	5	20
$\overline{\text{Tea}}$	75	5	80
	90	10	100

关联规则：Tea → Coffee

Confidence= P(Coffee|Tea) = 0.75

P(Coffee) = 0.9

⇒ 尽管置信度很高，但这个规则有误导性

⇒ P(Coffee|$\overline{\text{Tea}}$) = 0.9375

火龙果•整理
uml.org.cn

统计独立性

- 1000个学生
  - 600个会游泳 (S)
  - 700个会骑自行车 (B)
  - 420个会游泳和骑车 (S,B)

  - $P(S \wedge B) = 420/1000 = 0.42$
  - $P(S) \times P(B) = 0.6 \times 0.7 = 0.42$

  - $P(S \wedge B) = P(S) \times P(B)$ => 统计独立
  - $P(S \wedge B) > P(S) \times P(B)$ => 正相关
  - $P(S \wedge B) < P(S) \times P(B)$ => 负相关

火龙果•整理
uml.org.cn

基于统计的度量—兴趣度

- 兴趣度(Interest)

$$Interest(A,B)=P(AB)/(P(A)*P(B))$$

火龙果•整理
uml.org.cn

兴趣度(Interest)—示例

	Coffee	$\overline{\text{Coffee}}$	
Tea	15	5	20
$\overline{\text{Tea}}$	75	5	80
	90	10	100

Tea → Coffee

P(Coffee,Tea) = 0.15

P(Coffee) = 0.9

P(Tea) = 0.2

⇒ Interest = 0.15/(0.9*0.2)= 0.83

⇒(< 1，因此是负关联)

Tea → $\overline{\text{Coffee}}$

P($\overline{\text{Coffee}}$, Tea) = 0.05

P($\overline{\text{Coffee}}$) = 0.1

P(Tea) = 0.2

⇒Interest= 0.05/(0.1*0.2)= 2.5

⇒(> 1，因此是正关联)

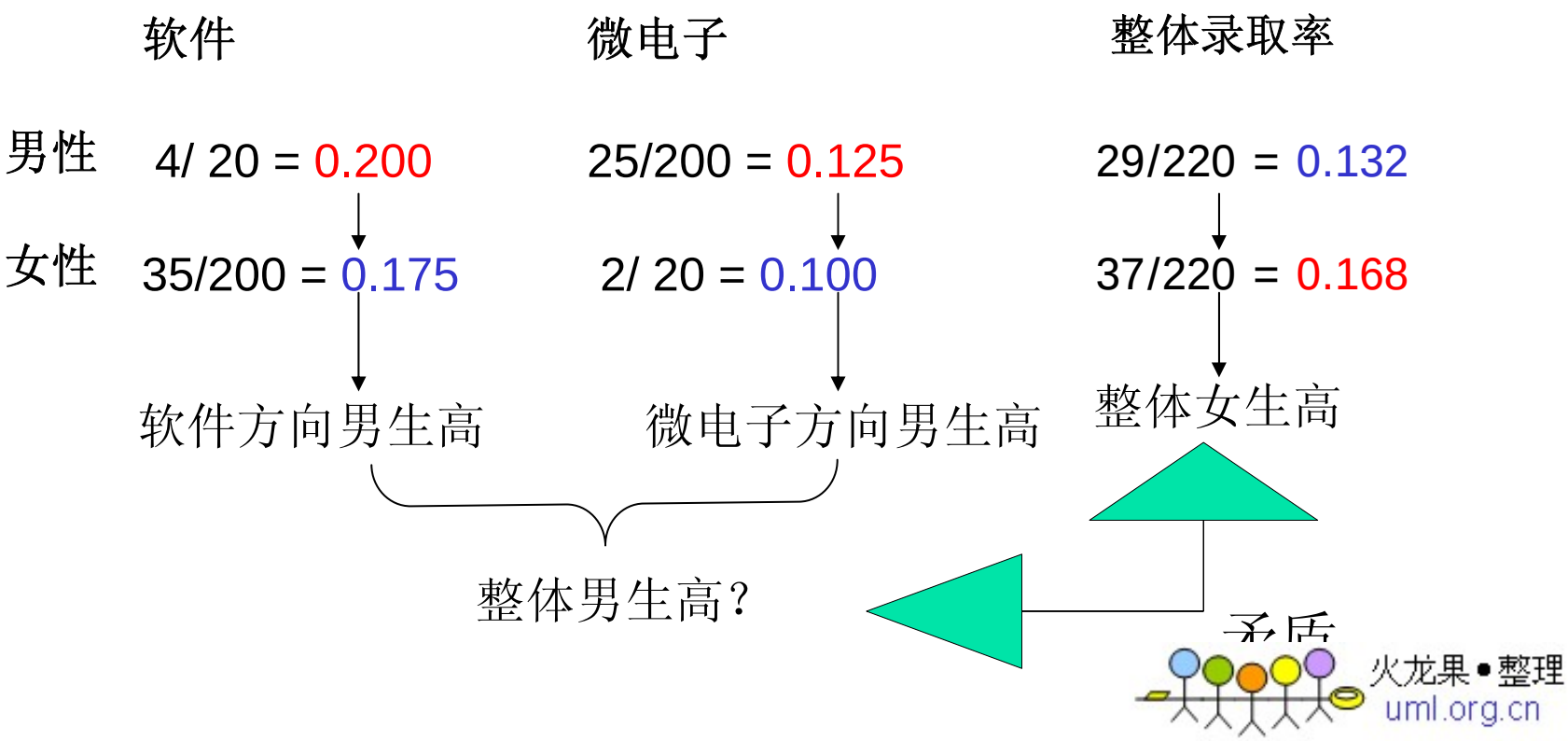
火龙果•整理
uml.org.cn

#	Measure	Formula
1	ϕ-coefficient	$\frac{P(A,B)-P(A)P(B)}{\sqrt{P(A)P(B)(1-P(A))(1-P(B))}}$
2	Goodman-Kruskal's (λ)	$\frac{\sum_j \max_k P(A_j,B_k)+\sum_k \max_j P(A_j,B_k)-\max_j P(A_j)-\max_k P(B_k)}{2-\max_j P(A_j)-\max_k P(B_k)}$
3	Odds ratio (α)	$\frac{P(A,B)P(\overline{A},\overline{B})}{P(A,\overline{B})P(\overline{A},B)}$
4	Yule's Q	$\frac{P(A,B)P(\overline{AB})-P(A,\overline{B})P(\overline{A},B)}{P(A,B)P(\overline{AB})+P(A,\overline{B})P(\overline{A},B)} = \frac{\alpha-1}{\alpha+1}$
5	Yule's Y	$\frac{\sqrt{P(A,B)P(\overline{AB})}-\sqrt{P(A,\overline{B})P(\overline{A},B)}}{\sqrt{P(A,B)P(\overline{AB})}+\sqrt{P(A,\overline{B})P(\overline{A},B)}} = \frac{\sqrt{\alpha}-1}{\sqrt{\alpha}+1}$
6	Kappa (κ)	$\frac{P(A,B)+P(\overline{A},\overline{B})-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A)P(B)-P(\overline{A})P(\overline{B})}$
7	Mutual Information (M)	$\frac{\sum_i \sum_j P(A_i,B_j)\log\frac{P(A_i,B_j)}{P(A_i)P(B_j)}}{\min(-\sum_i P(A_i)\log P(A_i), -\sum_j P(B_j)\log P(B_j))}$
8	J-Measure (J)	$\max\left(P(A,B)\log(\frac{P(B\|A)}{P(B)}) + P(A\overline{B})\log(\frac{P(\overline{B}\|A)}{P(\overline{B})}),\right.$ $\left. P(A,B)\log(\frac{P(A\|B)}{P(A)}) + P(\overline{A}B)\log(\frac{P(\overline{A}\|B)}{P(\overline{A})})\right)$
9	Gini index (G)	$\max\Big(P(A)[P(B\|A)^2+P(\overline{B}\|A)^2]+P(\overline{A})[P(B\|\overline{A})^2+P(\overline{B}\|\overline{A})^2]$ $-P(B)^2-P(\overline{B})^2,$ $P(B)[P(A\|B)^2+P(\overline{A}\|B)^2]+P(\overline{B})[P(A\|\overline{B})^2+P(\overline{A}\|\overline{B})^2]$ $-P(A)^2-P(\overline{A})^2\Big)$
10	Support (s)	$P(A,B)$
11	Confidence (c)	$\max(P(B\|A), P(A\|B))$
12	Laplace (L)	$\max\left(\frac{NP(A,B)+1}{NP(A)+2}, \frac{NP(A,B)+1}{NP(B)+2}\right)$
13	Conviction (V)	$\max\left(\frac{P(A)P(\overline{B})}{P(A\overline{B})}, \frac{P(B)P(\overline{A})}{P(B\overline{A})}\right)$
14	Interest (I)	$\frac{P(A,B)}{P(A)P(B)}$
15	cosine (IS)	$\frac{P(A,B)}{\sqrt{P(A)P(B)}}$
16	Piatetsky-Shapiro's (PS)	$P(A,B)-P(A)P(B)$
17	Certainty factor (F)	$\max\left(\frac{P(B\|A)-P(B)}{1-P(B)}, \frac{P(A\|B)-P(A)}{1-P(A)}\right)$
18	Added Value (AV)	$\max(P(B\|A)-P(B), P(A\|B)-P(A))$
19	Collective strength (S)	$\frac{P(A,B)+P(\overline{AB})}{P(A)P(B)+P(\overline{A})P(\overline{B})} \times \frac{1-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A,B)-P(\overline{A}$
20	Jaccard (ζ)	$\frac{P(A,B)}{P(A)+P(B)-P(A,B)}$
21	Klosgen (K)	$\sqrt{P(A,B)}\max(P(B\|A)-P(B), P(A\|$

火龙果•整理
uml.org.cn

Simpson's 悖论—例子1

软件与微电子学院录取率



火龙果•整理
uml.org.cn

Simpson's 悖论—例子2

顾客购买行为分析

Buy HDTV	Buy Exercise Machine		
	Yes	No	
Yes	99	81	180
No	54	66	120
	153	147	300

$$c(\{\text{HDTV} = \text{Yes}\} \rightarrow \{\text{Exercise Machine} = \text{Yes}\}) = 99/180 = 55\%$$
$$c(\{\text{HDTV} = \text{No}\} \rightarrow \{\text{Exercise Machine} = \text{Yes}\}) = 54/120 = 45\%$$

=> 买HDTV的顾客更有可能买健身器材

火龙果•整理
uml.org.cn

Simpson's 悖论

Customer Group	Buy HDTV	Buy Exercise Machine		Total
		Yes	No	
College Students	Yes	1	9	10
	No	4	30	34
Working Adult	Yes	98	72	170
	No	50	36	86

大学生：

$$c(\{\text{HDTV} = \text{Yes}\} \rightarrow \{\text{Exercise Machine} = \text{Yes}\}) = 1/10 = 10\%$$
$$c(\{\text{HDTV} = \text{No}\} \rightarrow \{\text{Exercise Machine} = \text{Yes}\}) = 4/34 = 11.8\%$$

职员：

$$c(\{\text{HDTV} = \text{Yes}\} \rightarrow \{\text{Exercise Machine} = \text{Yes}\}) = 98/170 = 57.7\%$$
$$c(\{\text{HDTV} = \text{No}\} \rightarrow \{\text{Exercise Machine} = \text{Yes}\})$$

火龙果•整理
uml.org.cn

Homework

- 思考一下Simpson’s 悖论的原因
- 尝试编写Apriori算法

火龙果•整理
uml.org.cn

下一讲

- 频繁模式挖掘高级话题

See you next time

火龙果•整理
uml.org.cn