Beijing Forest Studio

北京理工大学信息系统及安全对抗实验中心

关联规则分析相关算法介绍

李筱雅　硕士研究生

2018年09月22日

内容提要

- 背景简介
- 基本概念
- 算法原理
- 优劣分析
- 应用总结
- 参考文献

背景简介

- 预期收获
  - 了解关联规则的基本概念
  - 熟悉Apriori、GSP、SPADE算法原理
  - 了解关联规则算法的应用场景

背景简介

- Apriori算法是Agrawal等人于1993年为挖掘顾客交易数据库中项集之间的关联规则问题而提出的。
- 挖掘数据关联规则解决的问题：
  - 数据挖掘：购物篮分析——“尿布与啤酒”问题
  - 网页浏览偏好挖掘、入侵检测
  - 自然语言处理

基本概念

- 基本概念
  - 关联分析：在大规模数据集中寻找关系的非监督算法，两个目标：发现频繁项集(frequent item sets)和关联规则(association rules)。
  - 频繁集的度量标准：
    - 支持度：目标数据在整个事务数据库T中的出现的频率，假设T中有N条目标数据，支持度计算公式为：$Sup(X) = \frac{Sum(X)}{N}$
    - 置信度：规则 X ⇒ Y 在整个 T 中出现的频率。表示包含了 X 的条件下，还含有 Y 的事务占总事务的比例。计算公式为：$Conf(X \Rightarrow Y) = \frac{Sup(X \cup Y)}{Sup(X)}$

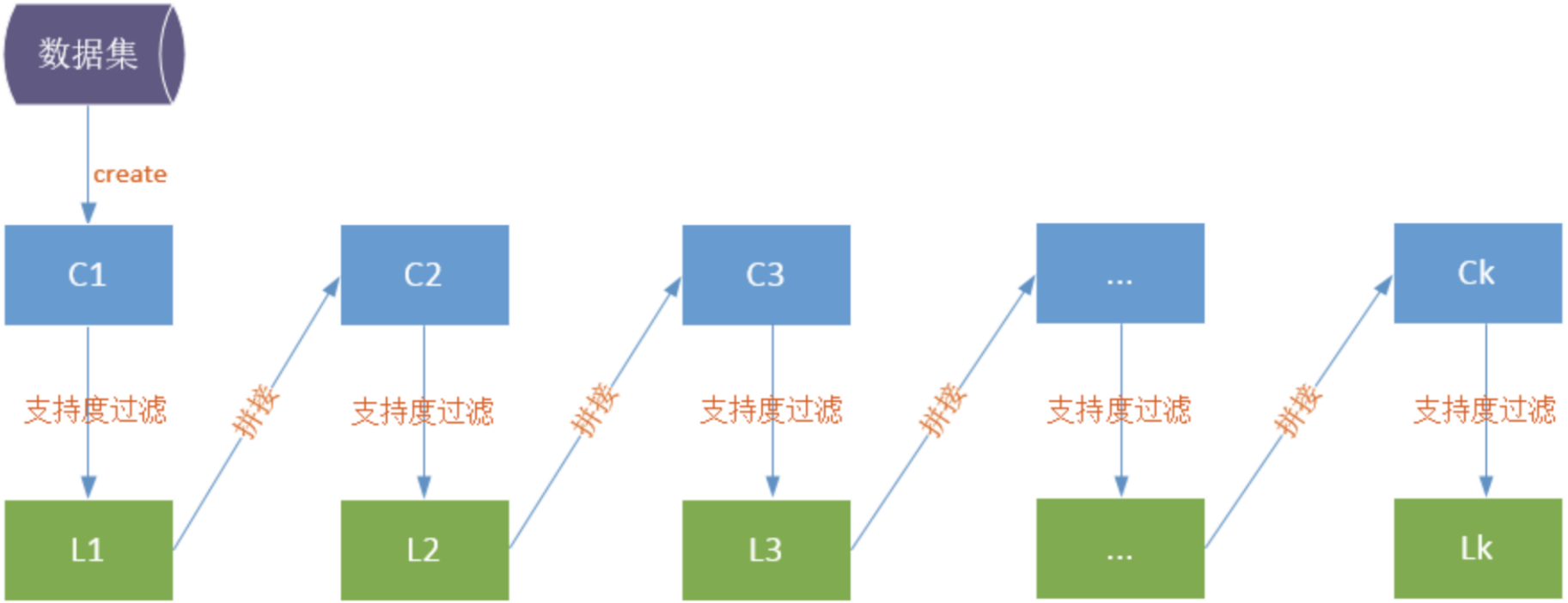
算法原理

T	关联规则分析
I	大规模数据集
P	1 发现频繁项集 2 发现关联规则
O	关联规则

P	如何挖掘项集之间的关联关系
C	数据之间存在一定的关联程度
D	候选集与频繁集的获取
L	ccf A类会议

算法原理

- Apriori算法过程



算法原理

- Apriori算法

交易记录	商品
0	豆奶，莴苣
1	莴苣，尿布，啤酒，甜菜
2	豆奶，尿布，啤酒，橙汁
3	莴苣，豆奶，尿布，啤酒
4	莴苣，豆奶，尿布，橙汁

– 寻找k项频繁集（设最小支持度为0.3）

C1		L1(一项频繁集)	
项集	支持度	项集	支持度
豆奶	4/5=0.8	豆奶	0.8
莴苣	0.8	莴苣	0.8
尿布	0.8	尿布	0.8
啤酒	3/5=0.6	啤酒	0.6
甜菜	1/5=0.2	甜菜	0.4
橙汁	2/5=0.4		

算法原理

C2		L2(二项频繁集)	
项集	支持度	项集	支持度
豆奶，莴苣	0.6	豆奶，莴苣	0.6
豆奶，尿布	0.6	豆奶，尿布	0.6
豆奶，啤酒	0.4	豆奶，啤酒	0.4
豆奶，橙汁	0.4	豆奶，橙汁	0.4
莴苣，尿布	0.6	莴苣，尿布	0.6
莴苣，啤酒	0.4	莴苣，啤酒	0.4
莴苣，橙汁	0.2	尿布，啤酒	0.6
尿布，啤酒	0.6	尿布，橙汁	0.4
尿布，橙汁	0.4		
啤酒，橙汁	0.2		

算法原理

C3		L3(三项频繁集)	
项集	支持度	项集	支持度
豆奶，莴苣，尿布	0.4	豆奶，莴苣，尿布	0.4
豆奶，莴苣，啤酒	0.2	豆奶，尿布，啤酒	0.4
豆奶，尿布，啤酒	0.4	豆奶，尿布，橙汁	0.4
豆奶，尿布，橙汁	0.4	莴苣，尿布，啤酒	0.4
莴苣，尿布，啤酒	0.4		

- ？【豆奶，莴苣，橙汁】
- 两两判断，如果两个项集的前K-1项相同，但第K项不同，则把两者拼接起来，组成备选项集

算法原理

- 发现关联规则
  - 置信度：$Conf(X \Rightarrow Y) = \frac{Sup(X \cup Y)}{Sup(X)}$

规则	支持度，置信度
尿布—>啤酒	（60%，75%）
啤酒—>尿布	（60%，100%）
。 。 。	。 。 。

  - 针对无序项目
  - GSP(Generalized Sequential Pattern)算法：基于Apriori，在序列中发现频繁序列模式
    - 两个步骤：1.自连接 2.剪枝

算法原理

- 自连接
  - 对于序列S1和S2，如果序列S1去掉第一项，与序列2去掉最后一项得到的序列相同，那么序列1和序列2就可以连接。把序列2的最后一项加入到序列1中，得到一个新的连接，即可以作为序列1和序列2连接的结果。
- 剪枝
  - 如果序列的支持度小于最小支持度，那么就会被剪掉
  - 如果序列是频繁序列，则它的所有子序列必定是频繁序列

算法原理

- GSP算法

顾客	Time（EID）	购物列表
A	1	电视机，电冰箱
A	2	冰箱清洁剂
A	3	冰箱贴
B	1	电冰箱
B	2	冰箱清洁剂
C	1	电冰箱
C	2	冰箱贴

  - 为方便，记电视机为I1，电冰箱为I2，冰箱清洁剂为I3，冰箱贴为I4，机顶盒为I5

算法原理

- 支持度：$Support(s)$ = s在n个序列中出现的次数
- 假定最小支持度为2
- 一项频繁序列

C1(一项候选序列)		L1(一项频繁序列)	
项集	支持度	项集	支持度
电视机I1	1	电冰箱I2	3
电冰箱I2	3	冰箱清洁剂I3	2
冰箱清洁剂I3	2	冰箱贴I4	2
冰箱贴I4	2		

算法原理

- 二项频繁序列

C2(二项候选序列)		L2(二项频繁序列)	
项集	支持度	项集	支持度
<I2,I2>	0	<I2,I3>	2
<I2,I3>	2	<I2,I4>	2
<I2,I4>	2		
<I3,I2>	0		
<I3,I3>	0		
<I3,I4>	1		
<I4,I2>	0		
<I4,I3>	0		
<I4,I4>	0		
(I2,I3)	0		
(I2,I4)	0		
(I3,I4)	0		

算法原理

- 时间约束：施加时限约束时，序列模式的每个元素都与一个时间窗口[l, u]相关联，使得有些序列不再支持候选模式。
- 最大跨越约束：序列中所允许的事件最晚和最早发生时间的最大时间差。

算法原理

- GSP的缺陷：
  - 每次计算序列的支持度时，都需要全表扫描数据集D
- SPADE算法：提出了ID_LIST的概念
- 对于电视机I1

顾客	Time(EID)
A	1

- 对于电冰箱I2

顾客	Time(EID)
A	1
B	1
C	1

算法原理

- 对于冰箱清洁剂I3

顾客	Time(EID)
A	2
B	2

- 对于冰箱贴I4

顾客	Time(EID)
A	3
C	2

C1(一项候选序列)		L1(一项频繁序列)	
项集	支持度	项集	支持度
电视机I1	1	电冰箱I2	3
电冰箱I2	3	冰箱清洁剂I3	2
冰箱清洁剂I3	2	冰箱贴I4	2
冰箱贴I4	2		

算法原理

- 求二项候选序列的ID_LIST(以<I2, I3>为例)

C2(二项候选序列)
项集
<I2,I2>
<I2,I3>
<I2,I4>
<I3,I2>
<I3,I3>
<I3,I4>
<I4,I2>
<I4,I3>
<I4,I4>
(I2,I3)
(I2,I4)
(I3,I4)

	顾客	Time(EID)
I2	A	1
	B	1
	C	1
I3	A	2
	B	2

- 由相同顾客购买I2先于I3的有:
  - (A, 1), (A, 2)
  - (B, 1), (B, 2)
- <I2, I3>的ID_LIST:

顾客	EID	EID
A	1	2
B	1	2

算法原理

- 同理可得<I2, I4>的ID_LIST:

顾客	EID	EID
A	1	3
C	1	2

- 结果:<I2, I3>, <I2, I4>

- 假设结果为<I2, I3>, <I3, I4>
  - 计算相同顾客在相同时间购买I3的次数作为支持度
  - 以此类推。。。

优劣分析

- Apriori算法
  - 优点：简单易实现
  - 缺点：多次扫描事务数据库，需要很大的I/O负载;可能产生庞大的候选集
- GSP算法
  - 优点：增加了新的扫描约束条件，有效的减少了需要扫描的候选序列数量，减少多余无用模式的产生。
  - 缺点：若序列数据库的规模较大，可能会产生大量的候选序列模式，需对数据库进行循环扫描。对序列模式长度较长时，由于产生短的序列模式规模太大

优劣分析

- SPADE算法
  - 优点：规避了对数据集的全表扫描问题，且ID_LIST的规模随剪枝不断缩小
  - 缺点：计算用户的支持度需要通过序列的Id_list进行统计，如果有大量的候选序列产生的时候,需要不断的进行连接操作, 降低了算法的执行效率
- 其他算法：FreeSpan算法、PrefixSpan算法、CloseSpan算法。。。

应用总结

- 数据挖掘
- 网络安全：
  - 网页浏览偏好挖掘
  - 入侵检测
  - 。。。
- 文本安全：
  - 关键词提取
  - 话题关联分析
  - 词性标注
  - 。。。

参考文献

[1] https://www.jianshu.com/p/7d459ace31ab关联挖掘算法

[2]Fast Algorithm For Mining Association Rule (Apriori-all 算法)

[3]https://en.wikipedia.org/wiki/Association_rule_learning关联规则学习

[4]https://blog.csdn.net/ztf312/article/details/50889238序列模型挖掘算法

[5] 李建俊. 字符序列模式挖掘算法的研究与应用[D].河北师范大学,2016.

知人者智，自知者明。
胜人者有力，自胜者
强。知足者富。强行
者有志。不失其所者
久。死而不亡者，寿。

谢谢！