Hadoop应用开发实战案例

第13周

法律声明

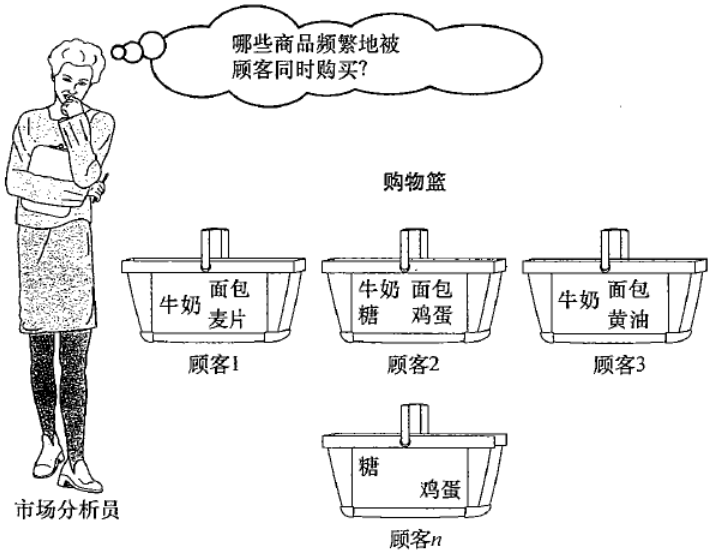
【声明】本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

课程详情访问炼数成金培训网站

http://edu.dataguru.cn

数据挖掘：关联规则挖掘

- 例子：购物篮分析



购物篮分析的应用

- 超市里的货架摆设设计
- 电子商务网站的套餐推荐

英国史5

当当价 ¥60.70 （6.9折） 钻石VIP专享折上9.5折

定　价 ¥88.00

评　论 ★★★★★ 97.4%推荐 156条

配送至 广东广州市海珠区 有货 运费说明 本商品提供礼品包装服务

今天(3月16日)可送达，请在9小时24分钟内下单并选择 "普通快递送货上门"

作　者 [英]大卫·休谟 著，刘仲敬 译

出 版 社 吉林出版集团有限责任公司

出版时间 2013-7-1

ＩＳＢＮ 9787553405445

所属分类 图书 > 历史 > 世界史 > 欧洲史

我要买 − 1 + 件

加入购物车 一键购买 收藏商品

分享到： 送积分 607 查看大图

批量购买入口>>

最佳拍档

英国史5 + 英国史6 ¥60.70 + 【乐扣当当旗舰店】650ml ¥39.60

1件商品组合购买

总当当价：¥60.70

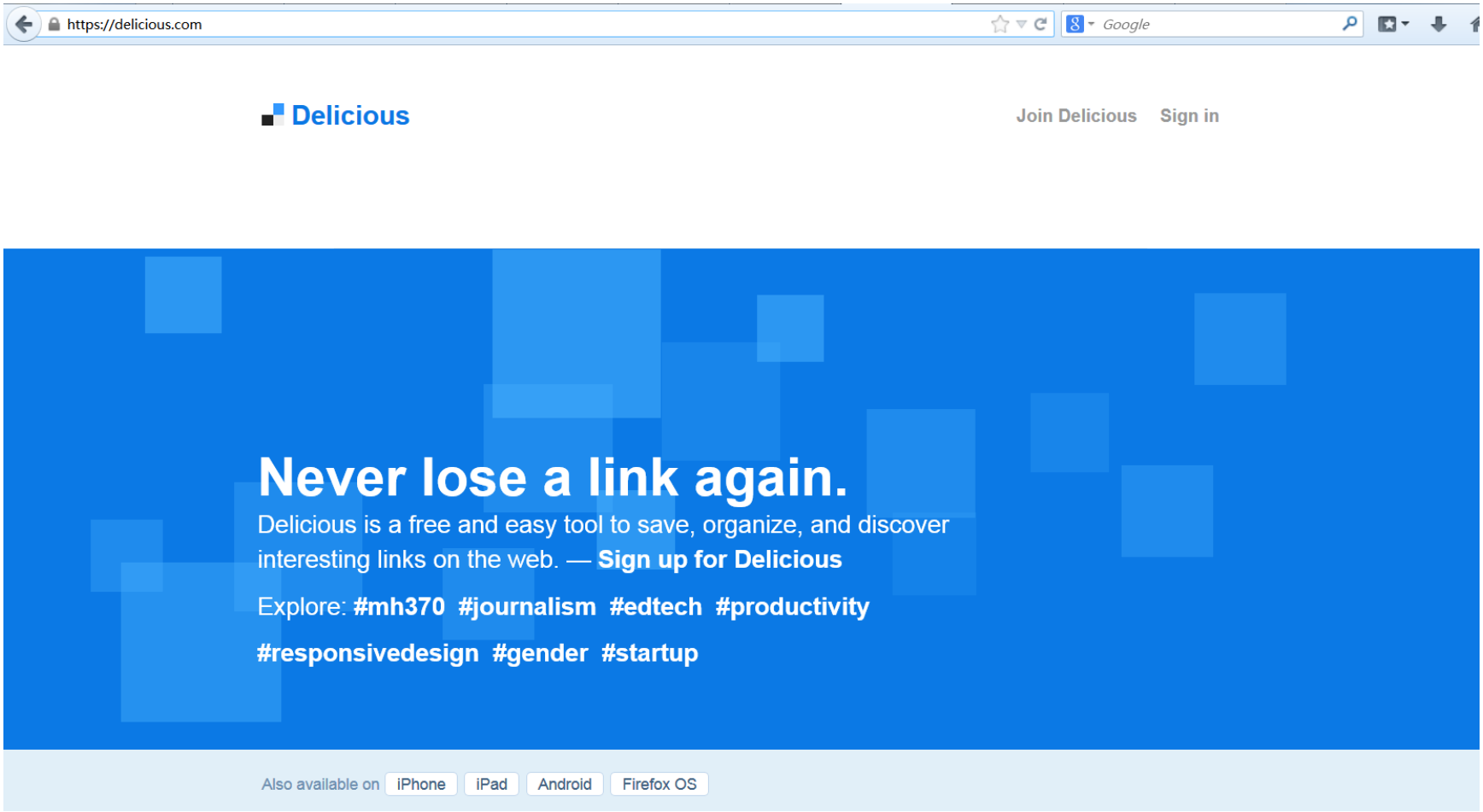
总定价：¥88.00

购买组合拍档

购物篮分析的应用

- 推荐系统：网站或节目的阅读/收听推荐

Delicious.com的研究



Delicious.com的研究

	TTD	WWD
URLs	802,939	802,739
Tags	1,021,107	1,021,107
Transactions	15,898,949	7,009,457
Total items	84,925,908	38,333,653

Table 1: Properties of the TTD (tag-tag) and WWD (webpage-webpage) transaction databases.

名词

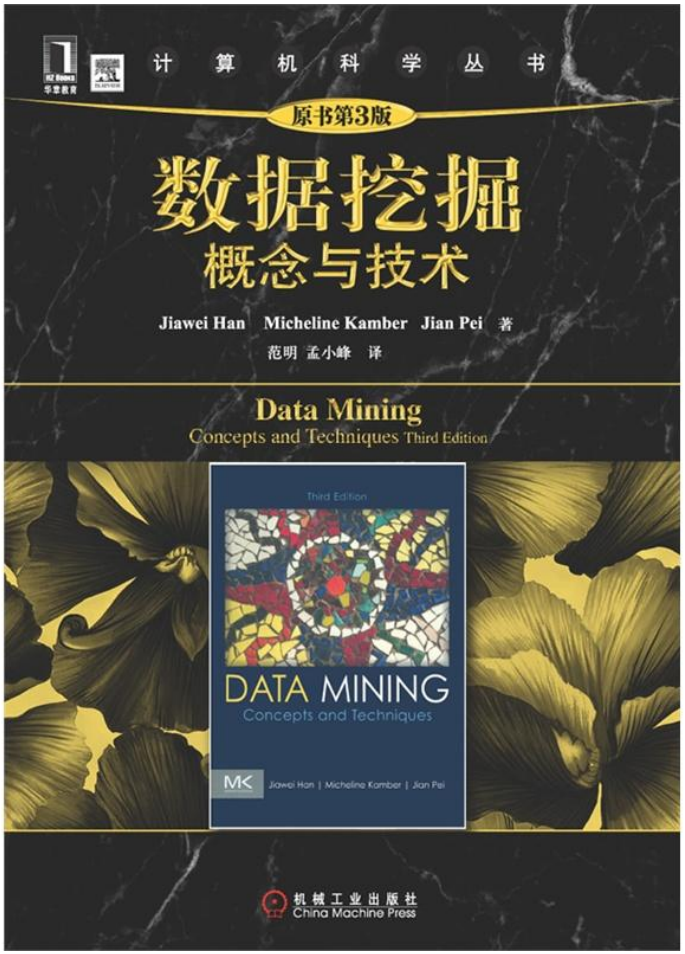
- 挖掘数据集：购物篮数据
- 频繁模式：频繁地出现在数据集中的模式，例如项集，子结构，子序列等
- 挖掘目标：频繁模式，频繁项集，关联规则等
- 关联规则：牛奶=>鸡蛋【支持度=2%，置信度=60%】
- 支持度：分析中的全部事务的2%同时购买了牛奶和鸡蛋
- 置信度：购买了牛奶的筒子有60%也购买了鸡蛋
- 最小支持度阈值和最小置信度阈值：由挖掘者或领域专家设定

名词

- 项集：项（商品）的集合
- k-项集：k个项组成的项集
- 频繁项集：满足最小支持度的项集，频繁k-项集一般记为L_k
- 强关联规则：满足最小支持度阈值和最小置信度阈值的规则

参考书

- 《Mahout in Action》一书的作者贡献了Mahout中频繁模式挖掘的代码
- 韩家炜是FPGrowth算法的创造者



关联规则挖掘： Apriori算法

- 两步过程：找出所有频繁项集；由频繁项集产生强关联规则
- 算法：Apriori
- 例子

表 6.1　AllElectronics 某分店的事务数据

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

Apriori算法的工作过程

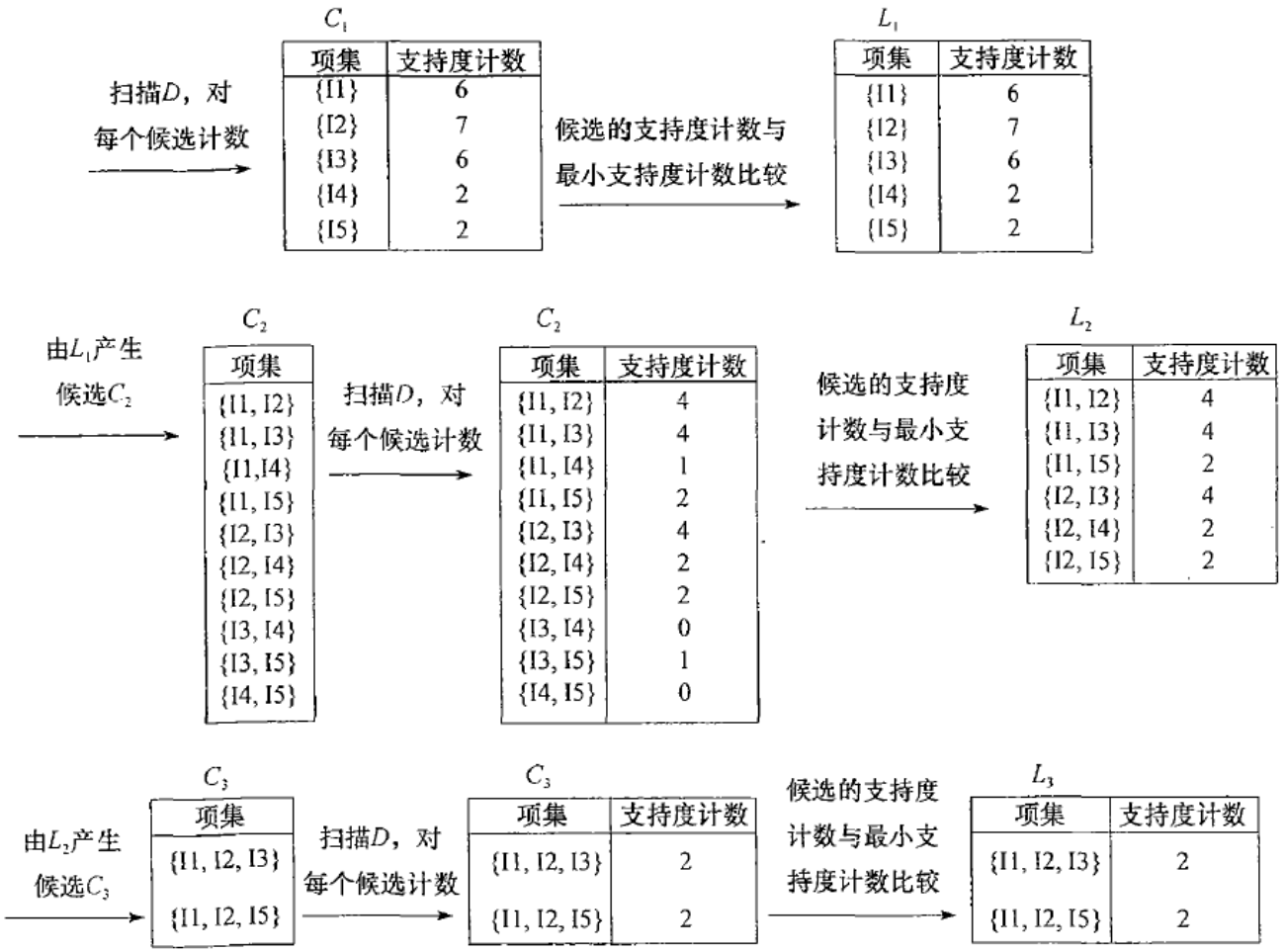


图 6.2　候选项集和频繁项集的产生，最小支持计数为 2

步骤说明

- 扫描D，对每个候选项计数，生成候选1-项集C1
- 定义最小支持度阈值为2，从C1生成频繁1-项集L1
- 通过L1xL1生成候选2-项集C2
- 扫描D，对C2里每个项计数，生成频繁2-项集L2
- 计算L3xL3，利用apriori性质：频繁项集的子集必然是频繁的，我们可以删去一部分项，从而得到C3，由C3再经过支持度计数生成L3
- 可见Apriori算法可以分成 **连接，剪枝** 两个步骤不断循环重复

步骤说明

（a）连接：$C_3=L_2 \bowtie L_2=\{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\}$

$\bowtie\{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\}$

$=\{\{I1, I2, I3\}, \{I1, I2, I5\}, \{I1, I3, I5\}, \{I2, I3, I4\}, \{I2, I3, I5\}, \{I2, I4, I5\}\}$

（b）使用先验性质剪枝：频繁项集的所有非空子集必须是频繁的。存在候选项集，其子集不是频繁的吗?

- {I1，I2，I3}的2项子集是{I1，I2}、{I1，I3}和{I2，I3}。{I1，I2，I3}的所有2项子集都是L_2的元素。因此，{I1，I2，I3}保留在C_3中。
- {I1，I2，I5}的2项子集是{I1，I2}、{I1，I5}和{I2，I5}。{I1，I2，I5}的所有2项子集都是L_2的元素。因此，{I1，I2，I5}保留在C_3中。
- {I1，I3，I5}的2项子集是{I1，I3}、{I1，I5}和{I3，I5}。{I3，I5}不是L_2的元素，因而不是频繁的。因此，从C_3中删除{I1，I3，I5}。
- {I2，I3，I4}的2项子集是{I2，I3}、{I2，I4}和{I3，I4}。{I3，I4}不是L_2的元素，因而不是频繁的。因此，从C_3中删除{I2，I3，I4}。
- {I2，I3，I5}的2项子集是{I2，I3}、{I2，I5}和{I3，I5}。{I3，I5}不是L_2的元素，因而不是频繁的。因此，从C_3中删除{I2，I3，I5}。
- {I2，I4，I5}的2项子集是{I2，I4}、{I2，I5}和{I4，I5}。{I4，I5}不是L_2的元素，因而不是频繁的。因此，从C_3中删除{I2，I3，I5}。

（c）因此，剪枝后$C_3=\{\{I1, I2, I3\}, \{I1, I2, I5\}\}$。

由频繁项集提取关联规则

- 例子：我们计算出频繁项集{I1,I2,I5}，能提取哪些规则？

I1^I2=>I5，由于{I1,I2,I5}出现了2次，{I1,I2}出现了4次，故置信度为2/4=50%

类似可以算出

$$\{I1,I2\}\Rightarrow I5,\quad confidence=2/4=50\%$$
$$\{I1,I5\}\Rightarrow I2,\quad confidence=2/2=100\%$$
$$\{I2,I5\}\Rightarrow I1,\quad confidence=2/2=100\%$$
$$I1\Rightarrow\{I2,I5\},\quad confidence=2/6=33\%$$
$$I2\Rightarrow\{I1,I5\},\quad confidence=2/7=29\%$$
$$I5\Rightarrow\{I1,I2\},\quad confidence=2/2=100\%$$

提高Apriori的效率

- 基于散列的算法
- 基于FP tree的算法

FP树

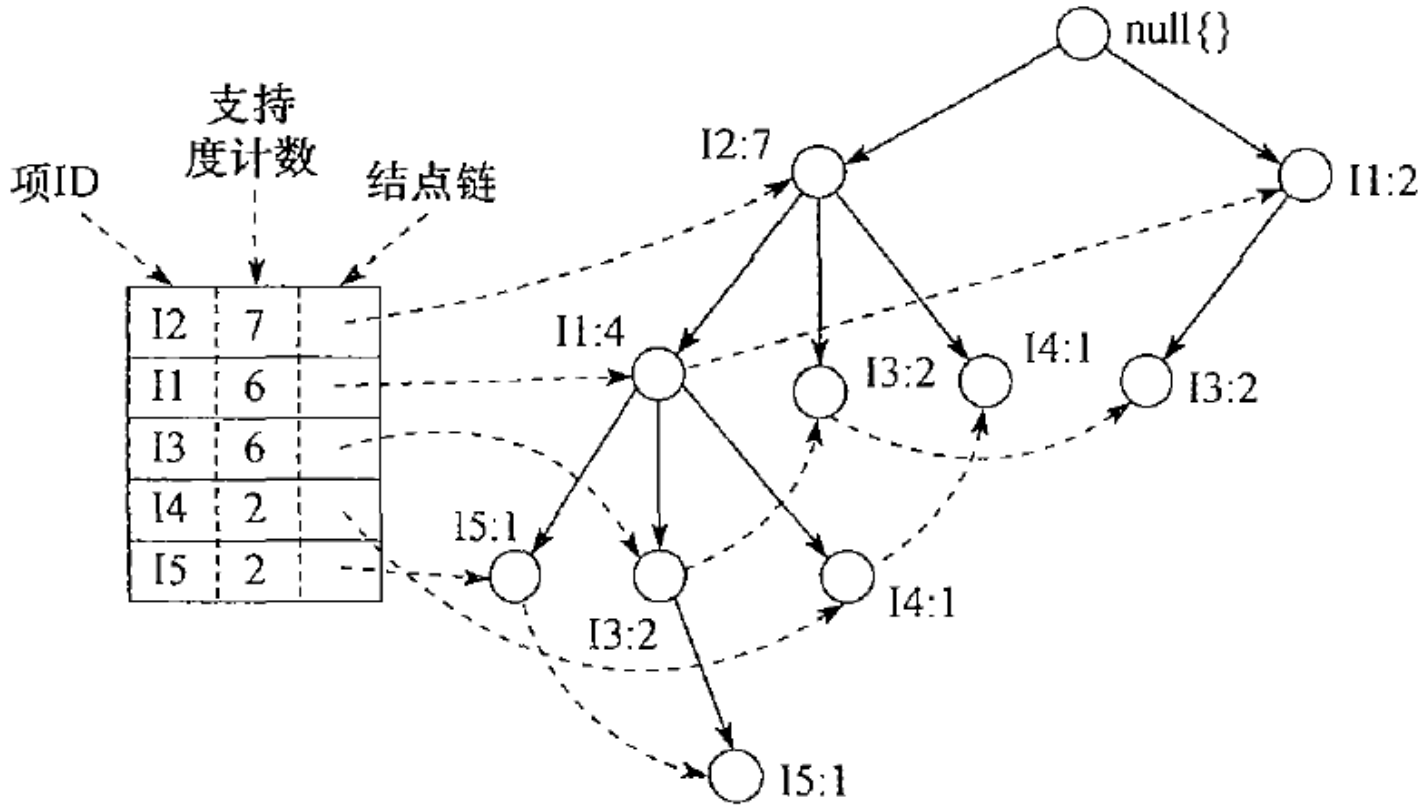


图 6.7　存放压缩的频繁模式信息的 FP 树

挖掘过程图示

表 6.2　通过创建条件（子）模式基挖掘 FP 树

项	条件模式基	条件 FP 树	产生的频繁模式
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}	⟨I2: 2, I1: 2⟩	{I2, I5: 2}、{I1, I5: 2}、{I2, I1, I5: 2}
I4	{{I2, I1: 1}, {I2: 1}}	⟨I2: 2⟩	{I2, I4: 2}
I3	{{I2, I1: 2}, {I2: 2}, {I1: 2}}	⟨I2: 4, I1: 2⟩, ⟨I1: 2⟩	{I2, I3: 4}、{I1, I3: 4}、{I2, I1, I3: 2}
I1	{{I2: 4}}	⟨I2: 4⟩	{I2, I1: 4}

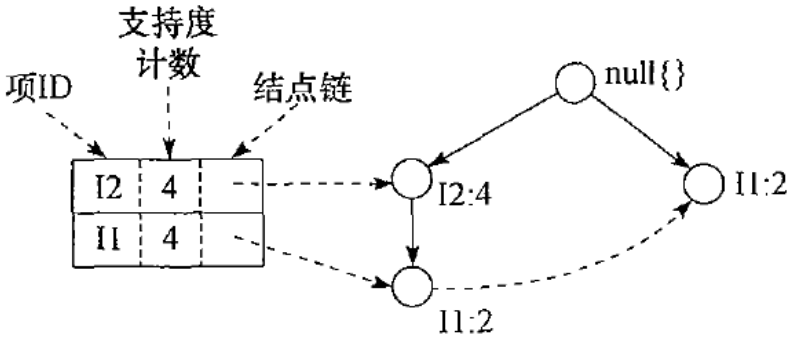


图 6.8　与条件结点 I3 相关联的条件 FP 树

FP-Growth算法

算法：**FP-Growth**。使用 FP 树，通过模式增长挖掘频繁模式。

输入：

- D：事务数据库。
- *min_ sup*：最小支持度阈值。

输出：频繁模式的完全集。

方法：

1. 按以下步骤构造 FP 树：
   (a)扫描事务数据库 D 一次。收集频繁项的集合 F 和它们的支持度计数。对 F 按支持度计数降序排序，结果为频繁项列表 L。
   (b) 创建 FP 树的根结点，以“null”标记它。对于 D 中每个事务 *Trans*，执行：
   选择 *Trans* 中的频繁项，并按 L 中的次序排序。设 *Trans* 排序后的频繁项列表为 $[p \mid P]$，其中 p 是第一个元素，而 P 是剩余元素的列表。调用 `insert_tree`$([p \mid P], T)$。该过程执行情况如下。如果 T 有子女 N 使得 $N.\ item\text{-}name = p.\ item\text{-}name$，则 N 的计数增加 1；否则，创建一个新结点 N，将其计数设置为 1，链接到它的父结点 T，并且通过结点链结构将其链接到具有相同 *item-name* 的结点。如果 P 非空，则递归地调用 `insert_tree`(P, N)。
2. FP 树的挖掘通过调用 `FP_growth`(*FP_tree*, *null*) 实现。该过程实现如下。

```
procedure FP_growth(Tree, α)
(1)if Tree 包含单个路径 P then
(2)for 路径 P 中结点的每个组合(记作 β)
(3)产生模式 β∪α,其支持度计数 support_count 等于 β 中结点的最小支持度计数;
(4)else for Tree 的头表中的每个 a_i {
(5)产生一个模式 β = a_i ∪ α,其支持度计数 support_count = a_i. support_count;
(6)构造 β 的条件模式基,然后构造 β 的条件 FP 树 Tree_β;
(7)if Tree_β ≠ ∅ then
(8)调用 FP_growth(Tree_β, β); }
```

PFP算法

- mahout提供了内存中的FPG和分布式的PFP两种算频繁项集的方法
- Parallel Frequent Pattern Mining ?
- Parallel FPGrowth ?
- https://cwiki.apache.org/confluence/display/MAHOUT/Parallel+Frequent+Pattern+Mining
- http://infolab.stanford.edu/~echang/recsys08-69.pdf

分布式FP-Growth

Map inputs (transactions) key="": value	Sorted transactions (with infrequent items eliminated)	Map outputs (conditional transactions) key: value
f a c d g i m p	f c a m p	p: f c a m m: f c a a: fc c: f
a b c f l m o	f c a b m	m: f c a b b: f c a a: fc c: f
b f h j o	f b	b: f
b c k s p	c b p	p: c b b: c
a f c e l p m n	f c a m p	p: f c a m m: f c a a: fc c: f

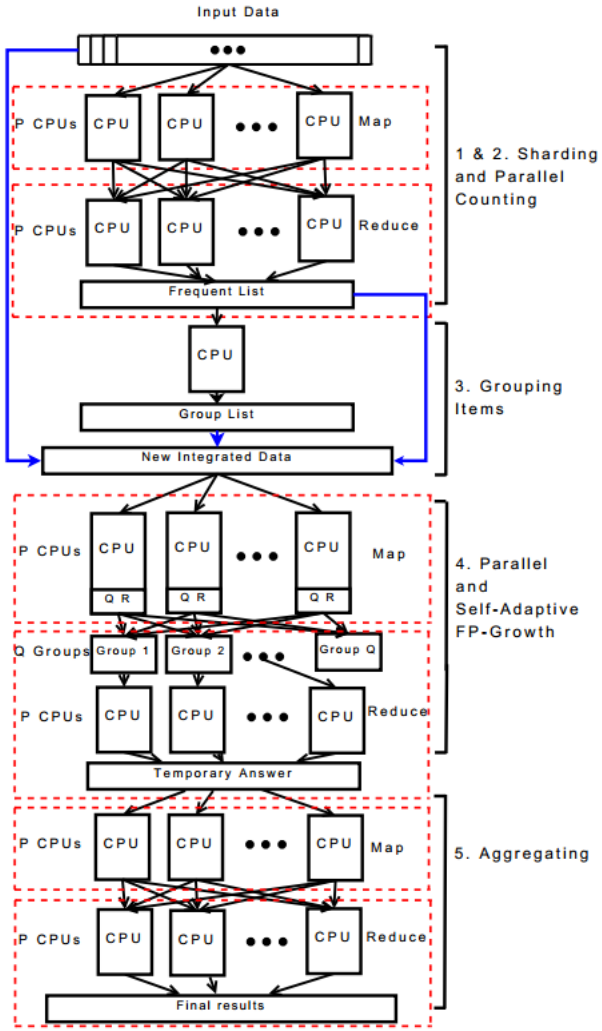
Reduce inputs (conditional databases) key: value		Conditional FP−trees
p:	{ f c a m / f c a m / c b }	{(c:3)} | p
m:	{ f c a / f c a / f c a b }	{ (f:3, c:3, a:3) } | m
b:	{ f c a / f / c }	{} | b
a:	{ f c / f c / f c }	{ (f:3, c:3) } | a
c:	{ f / f / f }	{ (f:3) } | c

Figure 1: A simple example of distributed FP-Growth.

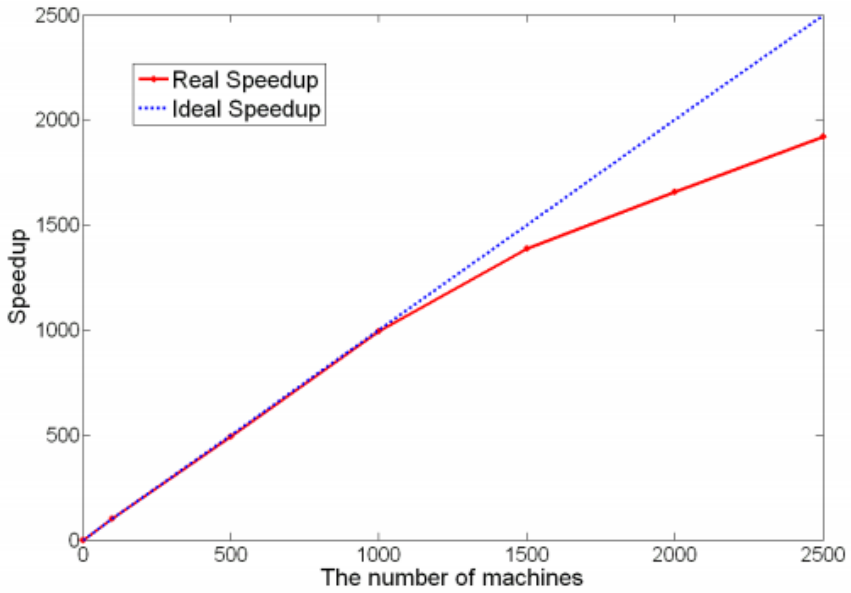
主要步骤

- 将数据集分片
- 计数，产生排序的F-List
- 将物品分组，产生G-List
- （PFP算法关键步骤）并行FP-Growth过程
- 聚合结果

PFP算法的五个阶段示意图



PFP计算能力评估



#. machines	#. groups	Time (sec)	Speedup
100	50000	27624	100.0
500	50000	5608	492.6
1000	50000	2785	991.9
1500	50000	1991	1387.4
2000	50000	1667	1657.1
2500	50000	1439	1919.7

Figure 4: The speedup of the PFP algorithm.

数据平台架构的重要性

- 算法只是案例的一个小部分
  - 数据挖掘算法的使用并不是实战案例最花时间的部分
  - 没有健康的数据生命流程, 挖掘算法将会变得没有意义
- 除了挖掘算法, 还有
  - 前期的数据准备
  - 后期的数据展现
- 一个好的数据平台能够持续让数据产生知识, 以至于利润
  - 好的数据平台能保障数据生命流程的健康

海量购物篮数据分析平台

数据数据提取层

- 分店定期导出数据
- 每次产生一个实例负责数据传输
- 因为各分店建立时间跨度较大，使用设备不一致，因此数据需要经过初步清洗

统一视图的后台数据库

- 必须考虑数据库的可扩展性
- Workload 并没有实时数据处理的部分，只需考虑离线分析
- 现成的有 Hbase 作为分布式数据库

分布式数据挖掘平台

- 用 Mahout 作为现成的数据挖掘平台
- 定期启动挖掘算法实例，捕获消费趋势变化
- 采用 Oozie 作为任务调度器

远程交互数据展现层

- 在网页前台展现数据分析结果，让一线销售人员掌握分析结果
- 交互式数据分析平台，多层次的展现分析结果
- 直接采用现成的可视化插件构建网页平台

商品关联分析 – 一个实例

- 实现海量购物篮数据分析平台的一个定期运行的关联挖掘算法

- 已知：
  - 购物篮数据库

- 要求：
  - 频繁项集的提取
  - 对关联关系的可视化展现

数据来源

- 阿里天梯比赛数据

字 段	字段说明	提取说明
user_id	用户标记	抽样&字段加密
Time	行为时间	精度到天级别&隐藏年份
action_type	用户对品牌的行为类型	包括点击、购买、加入购物车、收藏4种行为(点击：0 购买：1 收藏：2 购物车：3)
brand_id	品牌数字ID	抽样&字段加密

总体思路

用分布式 FP – Growth 算法挖掘频繁项集

利用频繁项集构建商品关系网络

对数据进行展现

Parallel FP - Growth

- Parallel FP – Growth
  - 能够在分布式计算平台 Hadoop 上实现
- Google 的几位科学家提出的算法
  - http://infolab.stanford.edu/~echang/recsys08-69.pdf
- Mahout 项目有这个算法的实现
  - https://mahout.apache.org/users/stuff/parallel-frequent-pattern-mining.html

用分布式 FP – Growth 算法挖掘频繁项集

- 在mahout上执行 FP – Growth

```
bin/mahout fpg \
     -i core/src/test/resources/retail.dat \
     -o patterns \
     -k 50 \
     -method mapreduce \
     -regex '[\ ]
```

- 参数解释：

参数	说明	可选值
--input / -i	输入路径	
--output / -o	输出路径	
--method / -method	计算方法（单机/分布式）	sequential | mapreduce
--splitterPattern / -regex	分隔符(正则表达式)	默认逗号分隔
--minSupport / -s	最小支持度阈值	默认为3

数据输出

- 数据结果是序列化的

```
bin/mahout seqdumper \
      -i patterns/frequentpatterns/part-?-00000 \
      -n 4
```

frequentPatterns-2.csv

```
1  10999=[([10999],6), ([10999, 11679, 16110, 16540],2)]
2  15018=[([15018],2)]
3  7208=[([7208],3)]
4  27060=[([27060],2)]
5  24274=[([24274],2)]
6  15019=[([15019],5)]
7  10893=[([10893],5)]
8  28065=[([28065],3)]
9  20578=[([20578],2)]
10 23656=[([23656],4)]
11 949=[([949],4)]
12 155=[([155],11), ([155, 21336, 7061],2), ([155, 22556],2), ([155,
```

数据转换

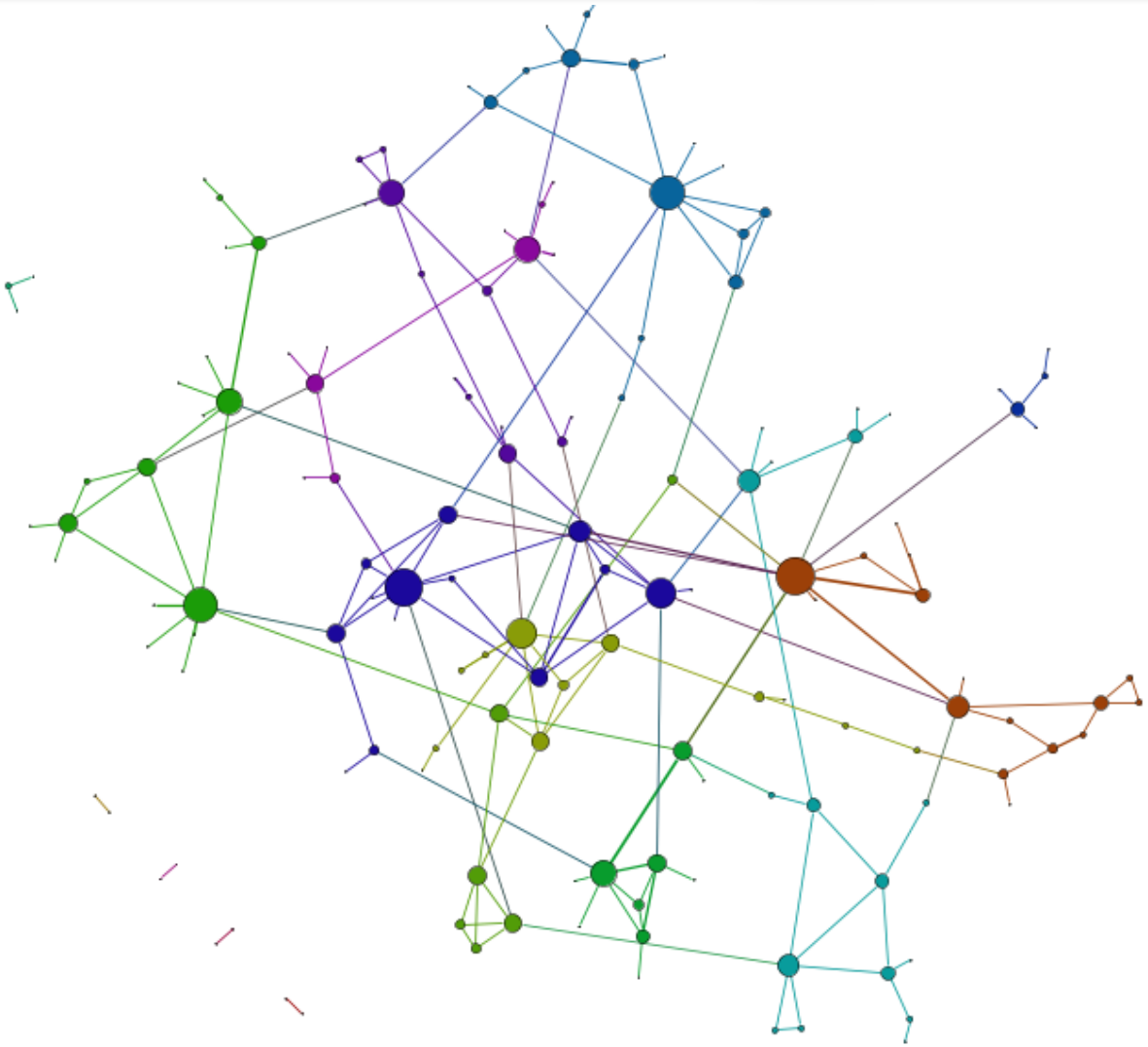
- 对数据进行转换, 生成一个边表, 用于Gephi展现

- 每一个节点代表一种商品
- 两个商品有边代表被多个买家同时购买
- 边的权值代表被同购的人数

edge.csv

	Source	Target	Weight
2	10702	23928	2
3	3913	11679	2
4	16331	27791	2
5	4571	21838	2
6	10702	18515	2
7	22043	22359	2
8	10014	18180	2
9	4571	19973	2
10	21336	23928	2
11	4807	15584	2
12	10999	16110	2
13	7096	23628	2
14	7096	21146	2
15	11080	24172	2
16	18075	26523	2
17	4949	8689	4

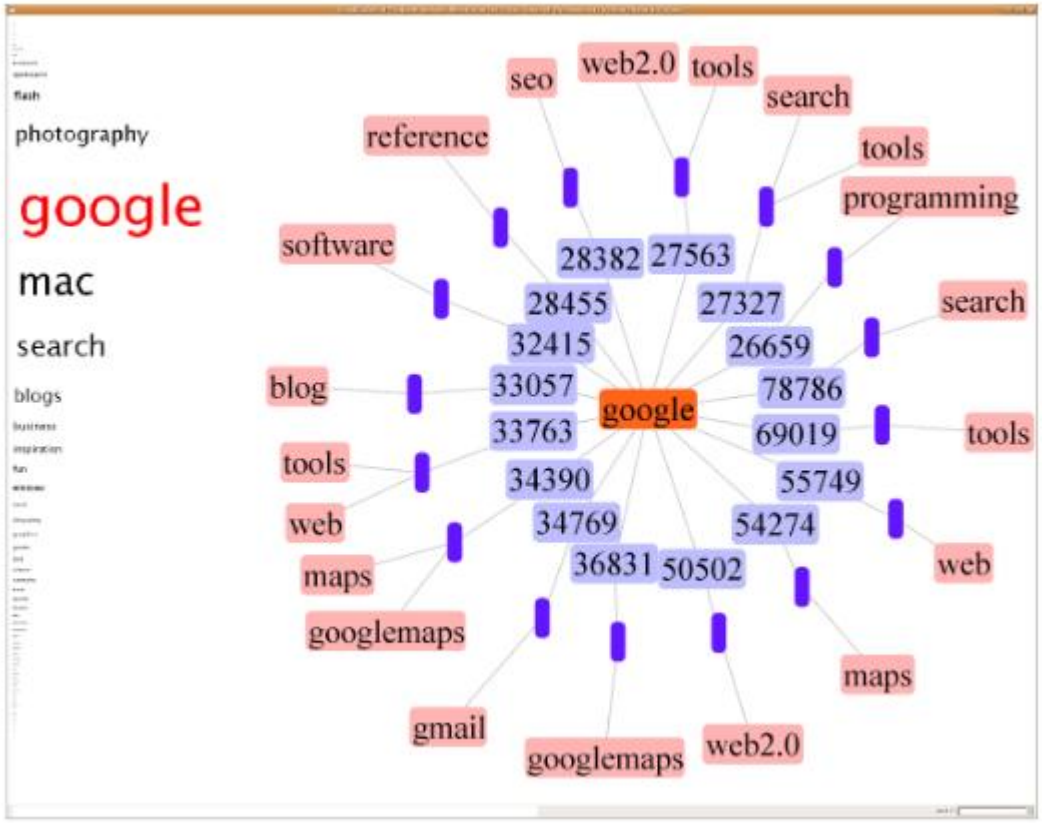
数据展现



一切皆可关联

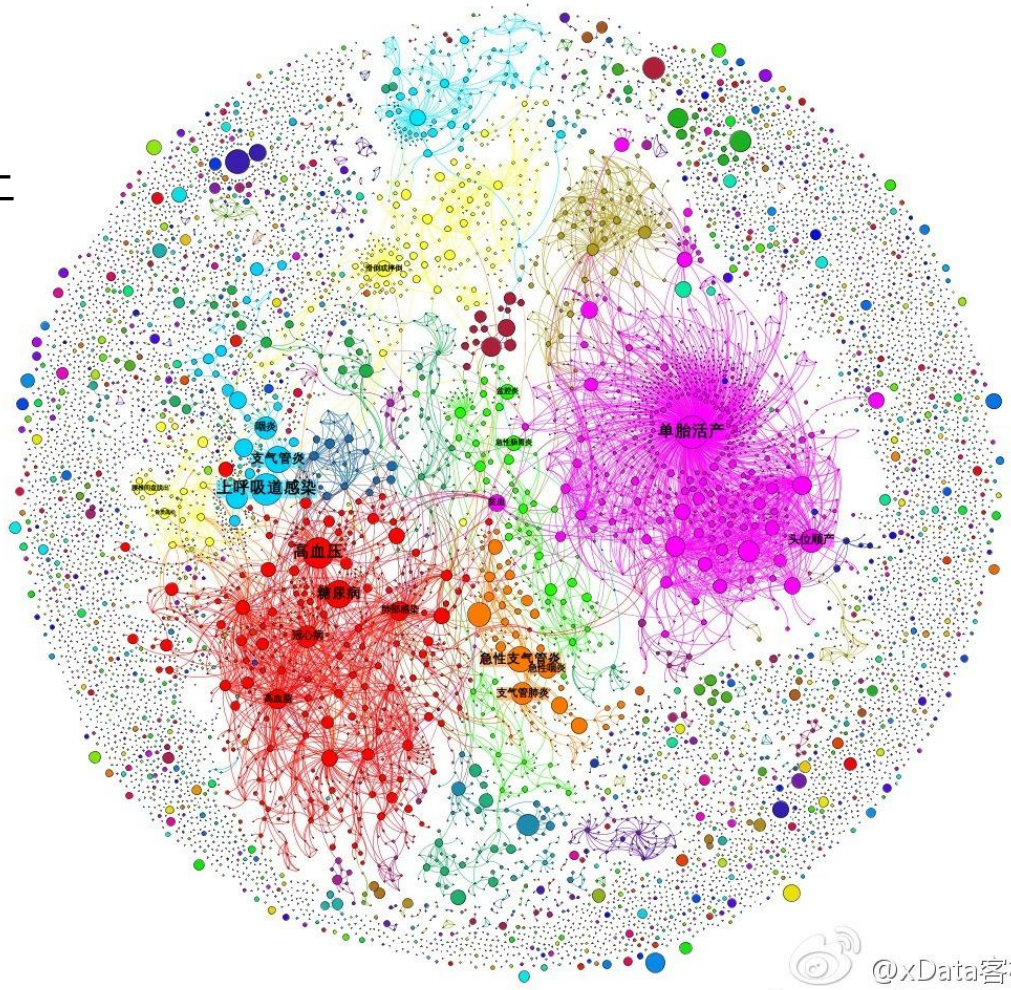
- 谷歌源论文展示的结果

- 网页标签关联



另一个实战案例 – 医疗大数据

- 海量病历数据
- 每一个点代表一种疾病诊断
- 边代表两个诊断出现在多个病历上



总结

- 本课讲解了
  - 关联规则的基础知识
    - 支持度
    - 置信度
  - 海量购物篮数据分析平台
    - 远程交互数据展现层
    - 分布式数据挖掘平台
    - 统一视图后台数据库
    - 分布式数据提取传输
  - 分布式 FP – Growth 算法挖掘频繁项集
    - Mahout 算法使用
    - 数据展现
    - 其他案例介绍

炼数成金逆向收费式网络课程

- Dataguru（炼数成金）是专业数据分析网站，提供教育，媒体，内容，社区，出版，数据分析业务等服务。我们的课程采用新兴的互联网教育形式，独创地发展了逆向收费式网络培训课程模式。既继承传统教育重学习氛围，重竞争压力的特点，同时又发挥互联网的威力打破时空限制，把天南地北志同道合的朋友组织在一起交流学习，使到原先孤立的学习个体组合成有组织的探索力量。并且把原先动辄成千上万的学习成本，直线下降至百元范围，造福大众。我们的目标是：低成本传播高价值知识，构架中国第一的网上知识流转阵地。
- 关于逆向收费式网络的详情，请看我们的培训网站 http://edu.dataguru.cn

Thanks

FAQ时间