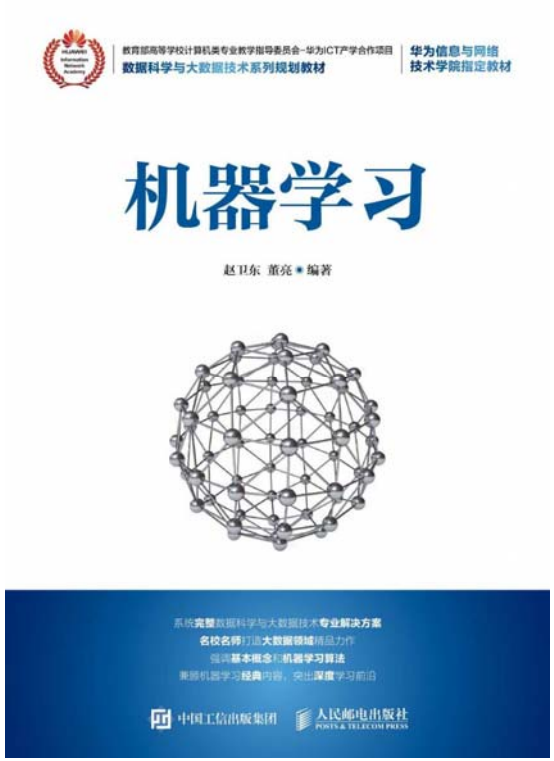


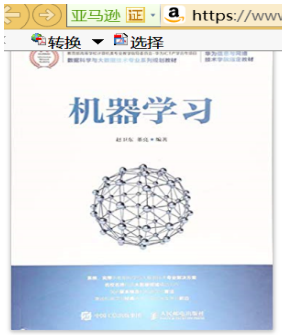
机器学习
关联规则

复旦大学 **赵卫东** 博士

wdzhao@fudan.edu.cn

关联规则

- 关联规则是反映物品与其他物品之间的关联性，常用于实体商店或在线电商的推荐系统：通过对顾客的购买记录数据进行关联规则挖掘，发现顾客群体的购买习惯的内在共性。早期的关联分析主要用于零售行业的购物行为分析，所以也称之为购物篮分析。需要注意的是关联关系并不意味着存在因果关系。关联规则分析中的关键概念包括支持度、置信度、提升度。在关联分析算法中，常见的有Apriori和FP增长算法。



查看全部 2 张图片

赵卫东 董亮 (作者)

2 条商品评论 | 分享

自营

显示所有 格式和版本

平装
￥51.20 prime

库存中仅剩 3 件（更多商品正在运送途中）

送达日期: 周日(2月17日) 送达，需要在**14小时35分钟以内选择快递送货上门确认您的订单（亚马逊Prime会员免费配送）**
(精确送达时间请于结账页面查询)

销售配送: 由亚马逊直接销售和发货。

全新品6 售价从 ￥51.20起

退换承诺: 此商品支持**7天无理由退货** 详情

精选好书

悦享春风读书季,赶快抢购吧：
精选好书每满100减30元，仅限亚马逊指定自营图书（不含进口原版书、进口繁体中文书、第三方卖家商品、电子书）。 点击查看详情>>

机器学习是人工智能的重要技术基础，涉及的内容十分广泛。本书内容涵盖了机器学习的基础知识，主要包括机器学习的概论、统计学习基础、分类、聚类、神经网络、贝叶斯网络、支持向量机、进化计算、文本分析等经典的机器学习理论知识，也包括用于大数据机器学习的分布式机器学习算法、深度学习和加强学习等高等级内容。此外，还介绍了机器学习的热门应用领域推荐技术，并给出了华为机器学习平台上的实验。本书深入浅出、内容全面、案例丰富，每章后都有习题和参考文献，便于学生巩固学习，适用于高

阅读更多

浏览此商品的顾客也同时浏览

深入浅出Python机器学习
段小手
平装
￥57.20 prime

机器学习基础:原理、算法与实践
袁梅宇
平装
￥58.60 prime

美团机器学习实践
美团算法团队
1
平装
￥66.10 prime

大数据技术与应用专业规划教材:数据挖掘实用案例分析
赵卫东
1
平装
￥47.80 prime

Python大战机器学习:数据科学家的第一个小目标
华校专
9
平装
￥53.71

机器学习实战
哈林顿 (Pete ...
166
平装
￥56.20 prime

Python 3 数据分析与机器学习实战
龙马高新教育
平装
￥52.83 prime

深入理解机器学习:从原理到算法
沙伊·沙莱夫-施瓦 ...
17
平装
￥63.20 prime

Python机器学习基础教程
[德]安德里亚斯· ...
4
平装
￥32.70

机器学习基
法与实践
袁梅宇
平装
￥55.20

购买此商品的顾客也同时购买

什么是关联挖掘?

- 关联规则挖掘：
  - 在交易数据、关系数据或其他信息载体中，查找存在于项目集合或对象集合之间的频繁模式、关联结构。
- 应用：
  - 购物篮分析、交叉销售、产品目录设计、 聚集和分类等。
- 举例：
  - 规则形式： “Body —>Head [support, confidence]”.
  - buys(x, “diapers”) —> buys(x, “beers”) [0.5%, 60%]
  - major(x, “CS”) ^ takes(x, “DB”) —> grade(x, “A”) [1%, 75%]

关联规则问题的形式化描述项目

- 定义1：集合I={i_1，i_2，…,i_m}为标识符的集合，其中m为正整数，i_k（k=1，2，…,m)称为项目。
- 项目是一个从具体问题中抽象出的一个概念。在超市的关联规则挖掘问题中，项目表示各种商品，如旅游鞋等。由于在超市的关联规则挖掘中并不关心顾客购买的商品数量和价格等，因此顾客的一次购物可以用该顾客所购买的所有商品的名称来表示，称为事务，所有事务的集合构成关联规则挖掘的数据集，称为事务数据库。

事务

- 定义2：关联规则挖掘的数据库记为D，事务数据库D中的每个元组称为事务。一条事务T是I中项目的集合。一条事务仅包含其涉及到的项目，而不包含项目的具体信息。在超级市场的关联规则挖掘问题中事务是顾客一次购物所购买的商品，但事务中并不包含这些商品的具体信息，如商品的数量、价格等。

事务数据集的矩阵表示

项目 / 事务	i_1	i_2	i_3	i_4	i_5	i_6
10	1	1	1	0	0	0
20	1	0	1	0	0	0
30	1	0	0	1	0	0
40	0	1	0	0	1	1

事务数据集

事务	购买商品（项集）
10	i_1，i_2，i_3
20	i_1，i_3
30	i_1，i_4
40	i_2，i_5，i_6

项目集

- 定义3：项目集是由I中项目构成的集合。若项目集包含的项目数为k，则称此项目集为k-项目集。
- 定义4：任意的项目集X和事务T若满足：$T \supseteq X$，则称事务T包含项目集X。
- 在超市的关联规则挖掘问题中项目集可以看成一个或多个商品的集合。若某顾客一次购买所对应的事务T包含项目集X，就说该顾客在这次购物中购买了项目集X中的所有商品。

频繁项目集

- 定义5：对任意的项目集X，若事务数据库D中ε%的事务包含项目集X，则项目集的支持率为ε，记为support（X）= ε，其中包含项目集X的事务数称为项目集X的频度，记为count（X）。若项目集X的支持率大于或等于用户指定的最小支持率（minsupport），则项目集X称为**频繁项目集**（或大项目集），否则项目集X为非频繁项目集（或小项目集）。如果数据库D中的事务数记为|D|，频繁项目集是至少被ε%x|D|条事务包含的项目集.

支持度和置信度

- 定义6:关联规则是形如X->Y的规则，其中X，Y为项目集且X∩Y=∅。
- 定义7：在数据库D中，若s%的事务包含X∪Y，则关联规则X->Y的支持度为s%；在数据库D中，若c%的包含项目集X的事务也包含项目集Y，则关联规则X->Y的置信度为c%:
- $p(Y|X) = p(XY)/p(X)$。
- 置信度反应了关联规则的可信度—购买了项目集X中的商品的顾客同时也购买了Y中商品的可能性有多大。

提升度

- 提升度用来判断规则是否有实际价值，描述的是对比不使用规则，使用规则可以提高多少。使用规则商品在购物车中出现的次数是否高于商品单独出现在购物车中的概率。大于1说明有效，小于1则无效。计算公式如下：

$$Lift(A \rightarrow B) = \frac{\text{Support}(A \cap B)}{Support(A) * Support(B)} \quad (1)$$

$$\text{Lift}(\text{A} \rightarrow \text{B}) = \frac{\text{Confidence}(A \rightarrow B)}{Support(B)} \quad (2)$$

- 例如，电商网站10月份有100万笔订单，购买面包30万笔，牛奶40万笔，同时购买两者的20万笔，面包、牛奶、面包和牛奶支持率依次为30%、40%、20%，所以提升度为1.667，大于1，所以牛奶面包规则是有提升效果的。

强关联规则

- 定义8：若关联规则X->Y的支持度和置信度分别大于或等于用户指定的最小支持率minsupport和最小置信度minconfidence，则称关联规则X->Y为**强关联规则**，否则称关联规则X->Y为弱关联规则。
- 关联规则挖掘的核心就是要找出事务数据库D中的所有强相关规则。

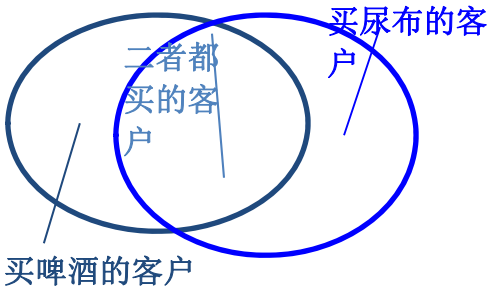
关联规则挖掘问题的分解

- 给定数据库D，关联规则的挖掘就是找出所有存在于数据库D中的强关联规则。因此整个关联规则挖掘过程可以分解为以下两个子问题：
- 找出所有的频繁项目集；
- 根据找到的频繁项目集导出所有的强关联规则。

强关联规则的产生

- 第一个子问题的求解，需要多次扫描数据库D，这意味着关联规则挖掘算法的效率将主要取决于数据库扫描、I/O操作和频繁项目集的计算上。因此如何迅速、高效地找出所有的频繁项目集是关联规则挖掘的中心问题
- 第二个子问题的求解比较容易，R. Agrawal等人已提出了有效的解决办法，具体过程如下：
- 对每个频繁项目集I，产生所有的非空真子集：对I的任意非空真真子集m，若support（I）/Support（m）≥minconfidence，则产生强关联规则m->(I-m)。

规则度量：支持度与可信度



交易ID	购买的商品
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

- 查找所有的规则 $X \& Y \Rightarrow Z$ 具有最小支持度和可信度
  - 支持度, *s*, 交易中包含{X 、 Y 、 Z}的可能性
  - 可信度, *c*, 包含{X 、 Y}的交易中也包含Z的条件概率

设最小支持度为50%, 最小可信度为50%, 则可得到

$A \Rightarrow C$ (50%, 66.6%)

$C \Rightarrow A$ (50%, 100%)

关联规则挖掘：路线图

- 布尔 vs. 定量 关联 (基于处理数据的类型)
  - buys(x, “SQLServer”) ^ buys(x, “DMBook”) → buys(x, “DBMiner”) [0.2%, 60%]
  - age(x, “30..39”) ^ income(x, “42..48K”) → buys(x, “PC”) [1%, 75%]
- 单维 vs. 多关联 (例子同上)
- 单层 vs. 多层分析
  - 哪个品牌的啤酒与那个牌子的尿布有关系?
- 各种扩展
  - 相关性、因果分析
    - 关联并不一定意味着相关或因果
  - 添加约束
    - 如哪些“小东西”的销售促发了“大家伙”的买卖?

关联规则挖掘例子

交易ID	购买商品
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

最小支持度 50%
最小置信度 50%

频繁项集	支持度
{A}	75%
{B}	50%
{C}	50%
{A,C}	50%

对于 $A \Rightarrow C$：

support = support({A 、 C}) = 50%

confidence = support({A 、 C})/support({A}) = 66.6%

Apriori的基本思想:

频繁项集的任何子集也一定是频繁的

如何生成候选集

- 假定 L_{k-1} 中的项按顺序排列
- 第一步: 自连接 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 $p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$
- 第二步: 修剪

  forall *itemsets c in* C_k do

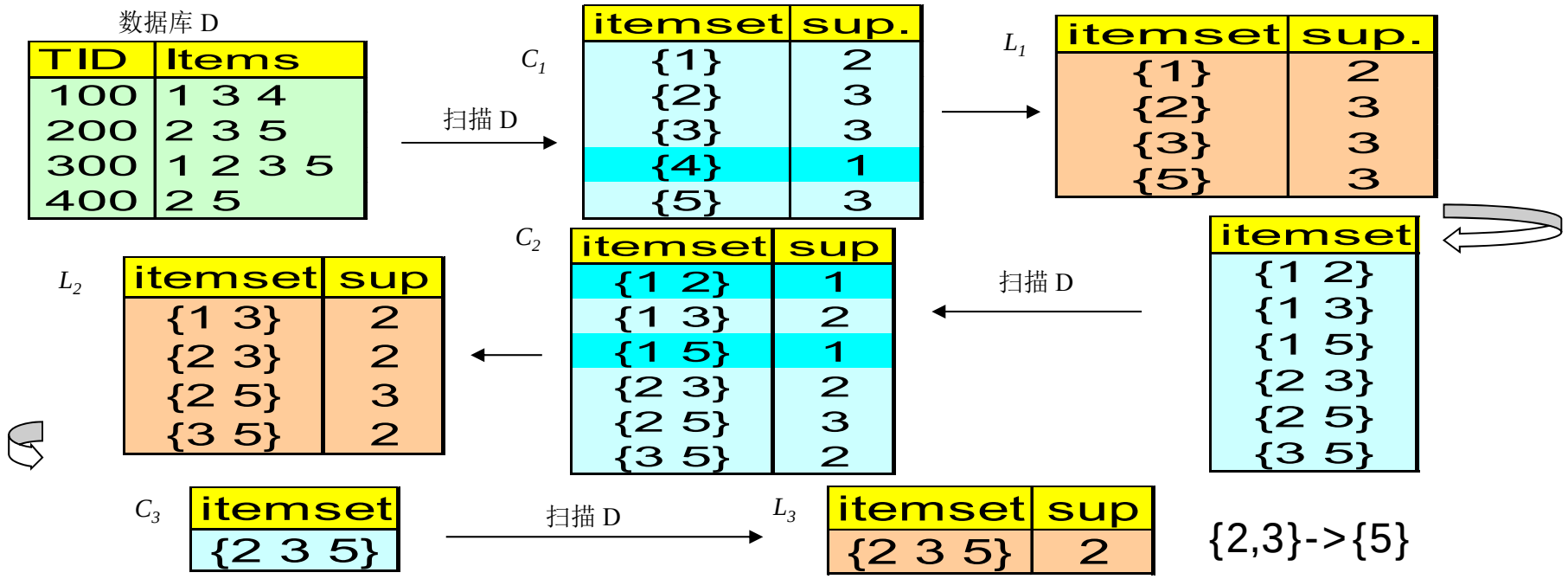
  forall *(k-1)-subsets s of c* do

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

生成候选集的例子

- L_3={*abc, abd, acd, ace, bcd*}
- 自连接 : L_3*L_3
  - *abc* 和 *abd* 得到 *abcd*
  - *acd* 和 *ace* 得到 *acde*
- 修剪:
  - *ade 不在 L_3 中，删除 acde*
- C_4={abcd}

Apriori算法例子



Apriori算法的Python实现

- Python的第三方库已经实现Apriori算法，可以使用pip3或pip安装，如pip3 install apyori，安装完成之后即可直接调用，示例如下：

```
from apyori import Apriori
transactions = [['bread', 'crisps'],
['milk', 'cheese'],
['bread', 'milk'],]
results = list(Apriori(transactions))
```

- 输出包含所有项集的集合，以及对应的支持度、置信度和提升度值，可以按照业务需求进行筛选和过滤。

基于关联分析的服装缺陷管理

- 服装制造通常采取流水线的方式生产，这是一个复杂的过程，其中牵涉多个部门。因而很难确定每个缺陷的责任部门，缺陷的根源无从查询。
- 一件产品身上的缺陷可能多达到上百处，这么大规模的缺陷信息缺少一套有效的机制管理和分析。低效率的缺陷识别导致服装行业的不良后果，例如客户满意度降低、返工成本高、生产周期长。

服装缺陷数据提取与预处理

- 把不同部门存储的有关制造流程和产品质量数据从不同的数据库中提取。因为不同部门使用的数据结构不一致，需要把这些数据预处理统一格式后存储到数据仓库中。i_1，i_2，i_3，i_4，i_5和i_6分别表示Broken stitches、Unraveling seams、Twisted leg、Poor colorfastness after being laundered、Sagging pockets和Re-stitched seams。

缺陷记录	产品编号	缺陷i_1	缺陷i_2	缺陷i_3	缺陷i_4	缺陷i_5	缺陷i_6
1	005	i_1				i_5	
2	028	i_1	i_2				i_6
3	032	i_1	i_2				i_6
4	058		i_2	i_3		i_5	
5	098						
6	105		i_2	i_3		i_5	
7	110	i_1	i_2	i_3		i_5	i_6
8	153			i_3	i_4		i_6
9	170	i_1	i_2				i_6
10	190		i_2	i_3		i_5	
11	197	i_1			i_4		
12	199	i_1	i_2				i_6

服装缺陷的部分关联规则

条件项	结果项	支持度(%)	置信度(%)
i_1i_2	i_6	41.70	100
i_1i_6	i_2	41.70	100
i_2i_6	i_1	41.70	100
i_1	i_2i_6	41.70	71.53
i_2	i_1i_6	41.70	62.52
i_6	i_1i_2	41.70	83.4
i_2i_3	i_5	33.30	100
i_2i_5	i_3	33.30	100
i_3i_5	i_2	33.30	100
i_2	i_3i_5	33.30	49.93
i_3	i_2i_5	33.30	79.86
i_5	i_2i_3	33.30	79.86

质量改善计划

- 质量管理小组就可以在一种缺陷存在的情况下，有效预测潜在的其他缺陷。
- 找出关联规则中缺陷同时存在的原因。例如缺陷$i_1 i_2$发生时，i_6也可能会同时发生，经过追查，根本原因为缝纫工人做工差。
- 质量管理人员应把更多的人力、设备等资源投入到解决这些问题的根源。

布尔型和数值型关联规则

- 根据处理的项目类别，关联规则可以分为布尔型和数值型。布尔型关联规则处理的项目都是离散的，它显示了这些变量之间的关系。例如性别=“女”→职业=“秘书”，是布尔型关联规则。而数值型关联规则可以和多维关联或多层关联规则结合起来。对数值型属性进行处理，参考连续属性离散化方法或统计方法把其进行分割，确定划分的区间个数和区间宽度。数值型关联规则中也可以包含可分类型变量。例如性别=“女”→平均收入>2300，这里的收入是数值类型，所以是一个数值型关联规则。又如，age(x，[30，39]) ^ income(x，[42，48]) →buys(x，“PC”) [1%，75%]。这里的项目用谓词表示，其中x是变量，泛指顾客，∧表示逻辑与。

挖掘多层关联规则

- 自上而下，深度优先的方法：
  - 先找高层的“强”规则：

    牛奶 → 面包 [20%, 60%].
  - 再找底层的“弱”规则：

    酸奶 → 黄面包 [6%, 50%].
- 多层关联规则的变种
  - 层次交叉的关联规则：

    酸奶 → 复旦面包房 黄面包
  - 不同种分层方法间的关联规则：

    酸奶 →复旦面包房面包

多层关联：冗余过滤

- 由于“祖先”关系的原因，有些规则可能是多余的。
- 例子
  - 牛奶 ⇒ 白面包 [support = 8%, confidence = 70%]
  - 酸奶 ⇒ 白面包 [support = 2%, confidence = 72%]
- 第一个规则是第二个规则的祖先
- 参考规则的祖先，如果它的支持度与“预期”的支持度近似的话，则这条规则是冗余的。

分布式并行Apriori算法（1）

- 第一阶段：生成局部的频繁项集
- 将大事务数据库D拆分成相同大小的n 个不相交的数据块，然后将它们发送到m个工作节点（m <=n）。
- 分别扫描每个数据块，并行使用Apriori 算法在每个数据块中产生局部频繁项集。

分布式并行Apriori算法（2）

- 第二阶段：生成全局频繁项集
- 将所有局部频繁项集进行合并，然后组合成全局候选的频繁项集合。局部频繁项集可能不是D的频繁项集，但在D中的任何频繁项集必然是局部频繁项集合中的一个子集。
- 对D进行再次扫描，计算每一个候选频繁项集的支持度，最终得到D的频繁项集。

分离关联规则

- 分离关联规则与一般的关联规则相似，只是在关联规则中出现项目的反转项，在购物篮分析中可发现不在一起购买的商品。例如购买牛奶的顾客一般不购买汽水。这里读者思考一下，如何发现分离关联规则？

FP增长算法

- 与Apriori算法不同，频繁模式增长（frequent pattern growth）算法，简称FP增长算法使用一种称为FP树的数据结构，并且采用分而治之的策略，无需产生候选频繁项集就能得到全部的频繁项集。

构造FP树

- FP树是事务数据库的压缩表示，每个事务都映射到FP树中的一条路径。不同的事务可能包含若干相同的项目，因此这些路径会有所重叠，使得事务数据能得到一定程度的压缩。FP增长算法挖掘频繁项集的过程如下：
- FP树构建过程包括以下三步：
  - 扫描数据集，得到所有频繁1项集的计数。然后删除支持度低于阈值的项，将1项频繁集放入项头表，并按照支持度降序排列。
  - 扫描数据集，将非频繁的1项集从原始数据集中剔除，并将事务按照支持度降序排列。
  - 读入排序后的事务集并按照排序顺序依次插入FP树中，排序靠前的节点是父节点，而靠后的是子节点。如果有共用的父节点，则对应的公用父节点计数加1。插入后，如果有新节点出现，则项头表对应的节点会通过节点链表链接到新节点。直到所有的数据都插入到FP树后，FP树的建立完成。

利用FP树产生频繁项集

- FP增长算法以自底向上的方式搜索FP树，由L的倒序开始，对每个1频繁项目构造条件FP树，然后递归地对该条件FP树进行挖掘。

项头表

商品(Item)	Pointer
鸡蛋：3	
面包：3	
啤酒：3	
牛奶：2	
黄油：1	

新事务集

事务(TID)	商品(Items)
10	鸡蛋、牛奶
20	鸡蛋、面包、啤酒
30	鸡蛋、面包、啤酒、牛奶
40	面包、啤酒