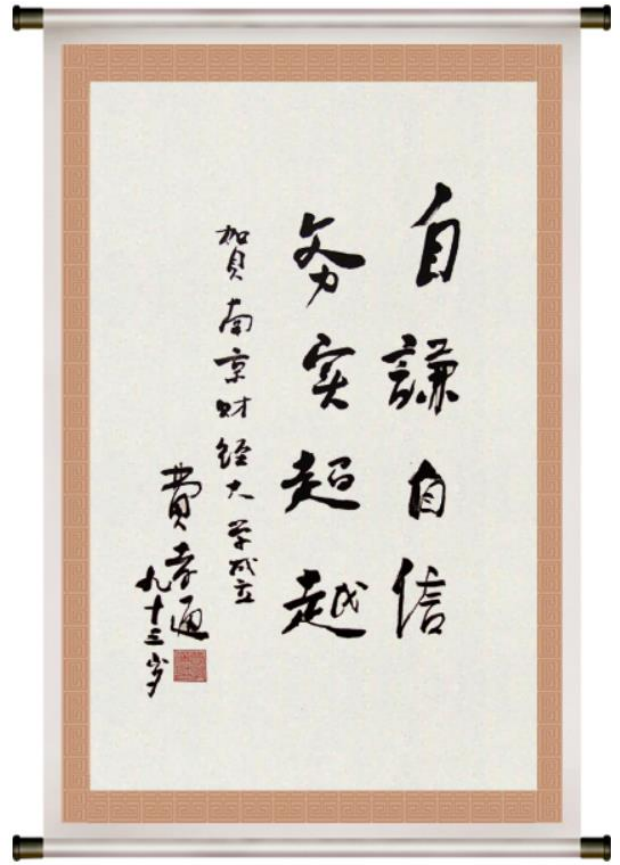
电子商务导论

第四章 商务数据挖掘

朱桂祥 (9120201070@nufe.edu.cn)
南京财经大学信息工程学院
江苏省电子商务重点实验室
电子商务信息处理国家级国际联合研究中心
电子商务交易技术国家地方联合工程实验室

南京财经大学校训



1.修身：

自谦、自信是从个人品德上来讲要保持谦虚之心，同时树立培养个人自信心；

2.立业：

务实、超越是从学习和行动上来讲要脚踏实地学习，本本分分做事，不断提升和超越自己；

格物致知，诚意正心，修身齐家治国平天下--选自儒学经典《礼记·大学》

格物致知，诚意正心，修身齐家治国平天下

“古之欲明明德于天下者，先治其国；欲治其国者，先齐其家；欲齐其家者，先修其身；欲修其身者，先正其心；欲正其心者，先诚其意；欲诚其意者，先致其知，致知在格物。物格而后知至，知至而后意诚，意诚而后心正，心正而后身修，身修而后家齐，家齐而后国治，国治而后天下平。”-----《礼记·大学》

- ✓ **格物致知**，即研究事物以求得知识。
- ✓ **诚意正心**，即要意念真诚端正心思。
- ✓ **修身**，即要提高个人的品德修养。
- ✓ **齐家**，即管理好自己的家庭（家族）。
- ✓ **治国**，即治理好国家的事务。
- ✓ **平天下**，即让天下太平。

习近平总书记在重要讲话中，对青年提出了希望和要求：**忠于祖国、忠于人民，立鸿鹄志、做奋斗者，求真学问、练真本领，知行合一、做实干家**。

目 录
Contents

一、关联规则挖掘

1. 背景

- ✓ 关联规则（Association Rules，AR）分析用于挖掘大规模数据集中有价值、有意义的联系，是数据挖掘领域的十大算法之一。
- ✓ 关联规则分析概念最早是1993年由Agrawal，Imielinski和Swami提出的，其主要研究目的是通过分析超市顾客购买行为的规律，发现连带购买商品，进而以此为依据来改善货架摆放方案(该分析称为购物篮分析)。
- ✓ 关联规则分析在购物篮数据分析、商品推荐营销、电子商务推广、生物信息学研究、医疗诊断咨询和航空电信等行业中都得到了广泛应用。
- ✓ Agrawal从数学及计算机算法角度出发，提出了商品关联关系的计算方法——Apriori算法。沃尔玛从上个世纪90年代尝试将Apriori算法引入到POS机数据分析中，获得了显著的业绩增长。

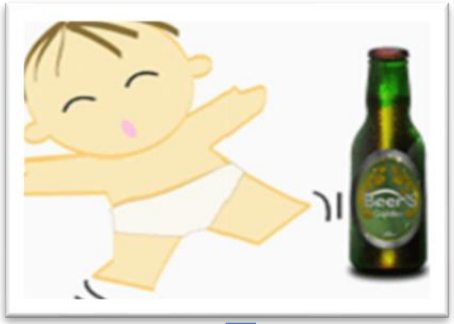
一、关联规则挖掘

- 啤酒与尿布

✓ 关于关联规则数据挖掘的应用，有一个流传甚广的案例：“啤酒与尿布”的故事。这个故事产生于20世纪90年代的美国沃尔玛超市中，沃尔玛的超市管理人员分析销售数据时发现了一个令人难于理解的现象：“啤酒”与“尿布”两件看上去毫无关系的商品会经常出现在同一个购物篮中。

✓ 这种独特的销售现象引起了管理人员的注意，经过调查发现，这种现象出现在年轻的父亲身上。在美国有婴儿的家庭中，一般是母亲在家中照看婴儿，年轻的父亲前去超市购买尿布。

✓ 父亲在购买尿布的同时，往往会顺便为自己购买啤酒，这样就会出现啤酒与尿布这两件看上去不相干的商品经常会出现在同一个购物篮的现象。如果这个年轻的父亲在卖场只能买到两件商品之一，则他很有可能会放弃购物而到另一家商店，直到可以一次同时买到啤酒与尿布为止。沃尔玛发现了这一独特的现象，开始在卖场尝试将啤酒与尿布摆放在相同的区域，让年轻的父亲可以同时找到这两件商品，并很快地完成购物；而沃尔玛超市也可以让这些客户一次购买两件商品，而不是一件，从而获得了更好的商品销售收入，这就是“啤酒与尿布”故事的由来。



一、关联规则挖掘

2. 事务与项集

- ✓ 关联规则分析研究的对象是事务(Transaction)，事务可以理解为一种商业行为。例如，超市顾客的购买行为是一种包含很多个商品购买的事务；网民的页面浏览行为是一种包含很多个页面访问的事务；一份保险公司的汽车保单是一种涵盖了若干个不同险种的事务。事务由序号和项集组成。序号是确定一个事务的唯一标识。
- ✓ 项目(Item)可以是一种商品、一个网页链接和一个险种。若干个项目的集合简称项集(Item Set)，若项集包含k个项目，则称该项集为k-项集。

一、关联规则挖掘

案例：

某个电商平台销售电商产品的APP当前有5项产品在售，为了分析顾客是否对这5项产品存在关联购买的行为，网站数据分析师抽取了9位顾客的购买记录（这里为了简化算法分析，选取的记录数远远低于真实场景），对他们的购买项目进行了统计。为了便于处理，使用数字代替{1：尿不湿、2：奶粉、3：啤酒、4：面包、5：电池}，得到电商产品购买记录清单如表4-1所示。

- ✓上表汇集了9个顾客的购买行为，每次购买行为称为一个事务，分别用A、B、C、D、E、F、G、H、I表示。
- ✓第一个事务A顾客一次性购买了3个产品，其项集为：{1：尿不湿、2：奶粉、5：电池}，是个3-项集。
- ✓本例中包含：
  - □ 5个2-项集
  - □ 3个3-项集
  - □ 1个4-项集

表 4-1 电商产品购买事务列表

事务ID	购买产品清单
A	1，2，5
B	2，4
C	2，3
D	1，2，4
E	1，3
F	2，3
G	1，3
H	1，2，3，5
I	1，2，3

一、关联规则挖掘

3.关联规则

- 关联规则研究的是事务集合内部的项集与项集之间的关系，这种关系有要有两种表现形式。
  - ✓ 频繁项集（Frequency Item Sets）：经常同时出现的一些项目的集合。
  - ✓ 关联规则（Association Rules）：项目集合A与项目集合B之间相互依存性和关联性。如果存在A→B的蕴涵式，意味着两种项目之间存在很强的某种联系。

一、关联规则挖掘

例子：

- 对于频繁项集的研究来说，就是要找出在上述商品集合{1：尿不湿、2：奶粉、3：啤酒、4：面包、5：电池}中，找出哪些“频繁”出现的子集组合。在表 4-1中，我们经过观察可以发现：产品组合{1、3}在事务E、G、H和I的购买清单中都出现了。也就意味着，买了产品1“尿不湿”的消费者，很有可能会购买产品3“啤酒”，反之亦然。
- 作为APP运营者，有理由相信，如果把{1 3}作为一个优惠套装，应该可以刺激两者的销量。
- “频繁项集”组合的项与项之间，存在着购买意向的“关联”。

表 4-1 电商产品购买事务列表

事务ID	购买产品清单
A	1，2，5
B	2，4
C	2，3
D	1，2，4
E	1，3
F	2，3
G	1，3
H	1，2，3，5
I	1，2，3

一、关联规则挖掘

4. 支持度和置信度

在进行算法分析之前，我们先了解两个重要的概念：支持度（Support）和置信度（Confidence）。支持度衡量规则在数据库中出现的频率，置信度衡量规则的强弱程度。

- ✓ 支持度：事务集D中，项目X和项目Y同时出现的概率，数学表达式为：

$$\text{Support}(X \rightarrow Y) = Pro（\text{X}\cup\text{Y}）=\frac{|\text{X}\cup\text{Y}|}{|\text{D}|}$$

例如：表 4-1中，事务个数为9，项目1和3同时发生的事务个数为4，则有：

$$\text{Support}(X \rightarrow Y) = \frac{4}{9}$$

即，同时购买项目1（“尿不湿”）和项目3（“啤酒”）的支持度为4/9（44.4%）。

一、关联规则挖掘

4. 支持度和置信度

- ✓ 置信度：即在出现项集X的事务集D中，项集Y也同时出现的概率，数学表达式为：

$$\text{Confidence}(X \rightarrow Y) = Pro（\text{Y}|\text{X}）=\frac{|\text{X}\cup\text{Y}|}{|X|}$$

例如：表 4-1中，1发生的事务数量为6，1和3同时发生的事务个数为4，则有：

$$\text{Confidence}(1 \rightarrow 3) = \frac{4}{6}$$

即，购买项目1（“尿不湿”）同时购买项目3（“啤酒”）的置信度为4/6 （66.7%），置信度越高，说明X出时，Y也同时出现的可能性越高。

一、关联规则挖掘

5. 最小支持度和最小置信度

- ✓ 最小支持度是用户定义的衡量支持度的一个阈值，表示项目集在统计意义上的最低重要性。
- ✓ 最小置信度是用户定义的衡量置信度的一个阈值，表示关联规则的最低可靠性。
- ✓ 同时满足最小支持度阈值和最小置信度阈值的规则称作强关联规则，否则称作为弱关联规则。
- ✓ 数据挖掘主要就是对强规则的挖掘，通俗地讲，就是要达到一定的门槛，我们才将这种现象纳入考虑范围。
- ✓ 如果项集满足最小支持度，则称之为频繁项集（Frequent Itemset）。

一、关联规则挖掘

6. 常见的关联规则挖掘算法

- ✓ **Apriori**

Apriori算法是第一个关联规则挖掘算法，也是最经典的算法。它利用逐层搜索的迭代方法找出数据库中项集的关系，以形成规则，其过程由连接（类矩阵运算）与剪枝（去掉那些没必要的中间结果）组成。

- ✓ **FP-Growth**

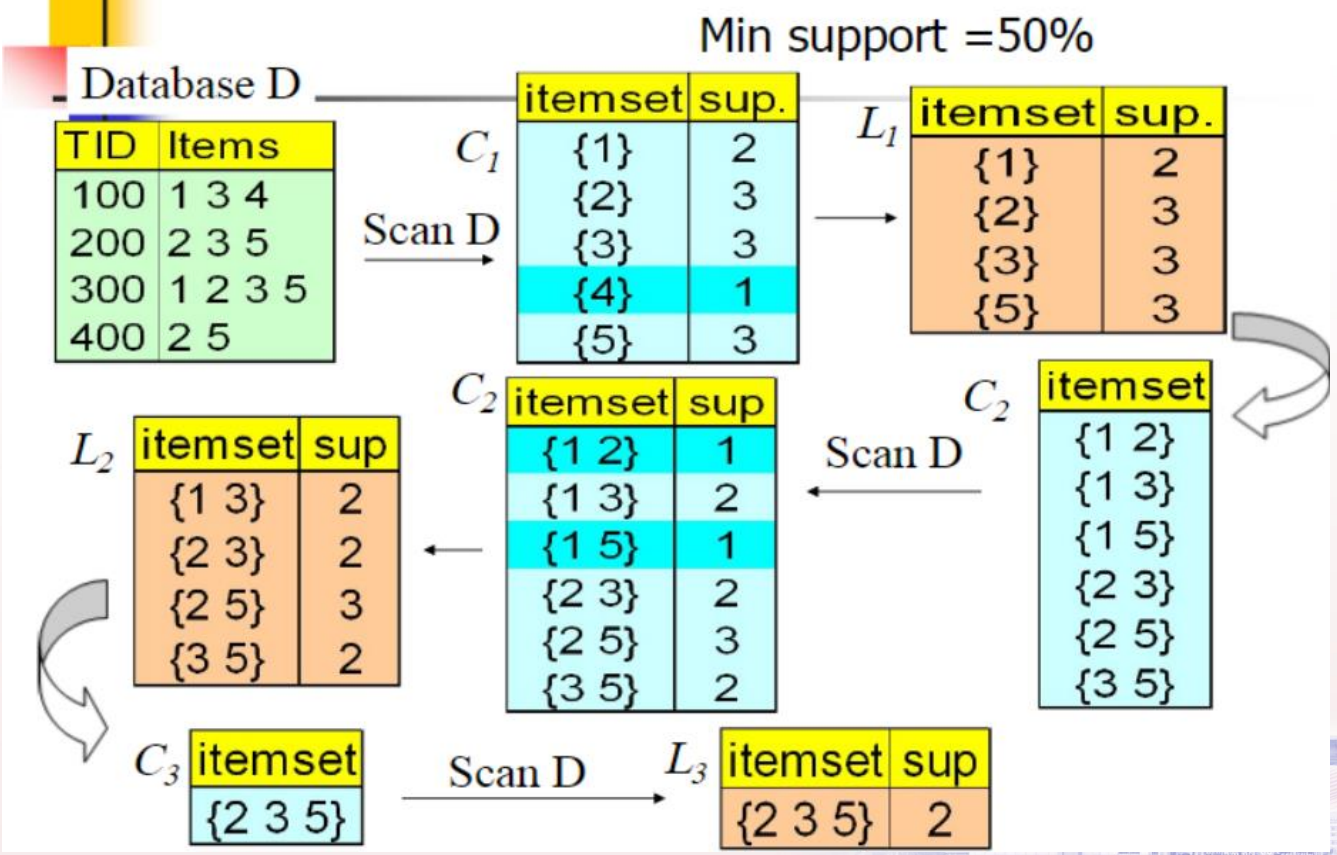
FP-Growth算法是Jiewei Han等人在2000年提出的关联分析算法，它采取如下分治策略：将提供频繁项集的数据库压缩到一棵频繁模式树（FP-tree），但仍保留项集关联信息。

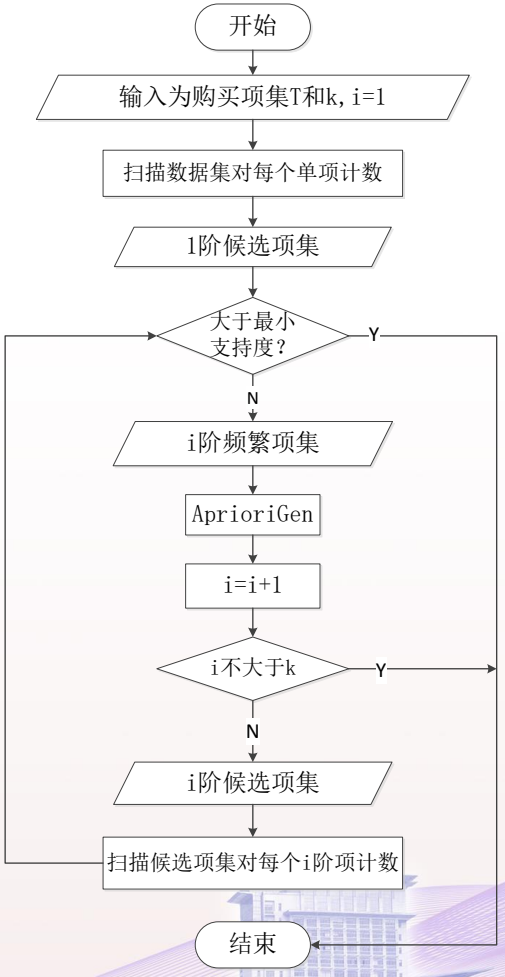
- ✓ **Eclat (equivalence class transformation)**

Eclat是一种深度优先算法,采用垂直数据表示形式,在概念格理论的基础上利用基于前缀的等价关系将搜索空间(概念格)划分为较小的子空间(子概念格)。

一、关联规则挖掘

7. Apriori算法原理





一、关联规则挖掘

8. Apriori算法Python实现

✓ **逐步采用Python实现关联规则挖掘算法：**

根据以上分析，针对表 4-1的电商产品购买例子，为了方便程序实现，我们将商品用数字来代替：{1：尿不湿、2：啤酒、3：奶粉、4：面包、5：电池}。

In[1]:

```
1 #Apriori 算法
2 #构建数据集
3 def loadDataSet():
4 return [[1, 2, 5], [2, 4], [2, 3], [1, 2, 4], [1, 3], [2, 3], [1, 3], [1, 2, 3, 5], [1, 2, 3]]
5
6 D=loadDataSet()
7 print(D)
```

Out: [[1, 2, 5], [2, 4], [2, 3], [1, 2, 4], [1, 3], [2, 3], [1, 3], [1, 2, 3, 5], [1, 2, 3]]

In[1]:主要设计了一个 loadDataSet 函数，用于读入事务集数据。

In[1]:的第 7 行加入了输出语句，用于输出读入的事务集数据。

一、关联规则挖掘

8. Apriori算法Python实现

```
1  #扫描 loadDataSet 返回的事务集，将所有的单项放入 C 中，构造出单项集
2  #参数：dataset-List 列表，事务集
3  #返回值：List-冻结后的单项集列表
4  def createC1(dataSet):
5      C = []
6      for transaction in dataSet:
7          for item in transaction:
8              if [ item ] not in C:
9                  C.append([item])
10     C.sort()
11 #调用 frozenset 函数将单项集列表冻结,返回冻结单项集列表
12     return list(map(frozenset, C))
13
14 C1=createC1(D)
15 print(C1)
```

设计了createC1函数。该函数接收事务集数据，从中提取出所有的单项，返回的结果是这些单项构成的集合。对于Python来说，list列表是一个可变集合，为了对这些单项列表进行进一步的组合和查询操作，需要将它转变为一个冻结（不可变）集合，因此，在函数的末尾进行了映射和转换：list(map(frozenset, C))。

第14-15行加入输出语句，输出的是从事务集中取出的单项集列表，其中每一个元素都是冻结的set集合。

一、关联规则挖掘

8. Apriori算法Python实现

```
#根据 minSupport 参数设定的最小支持度，
#计算 Ck 的项集在原始记录 D 中的支持度，
#返回满足最小支持度的项集集合，以及所有项集支持度的字典
#参数：D-List 列表，事务集
#       Ck-List 列表，候选项集
#       minSupport-浮点数，最小支持度阈值
#返回值：retList-List 列表，满足最小支持度要求的项集
#          supportData-Dict 字典，项集对应的支持度
def scanD(D, Ck, minSupport):
    ssCnt = {}
# 对于每一个候选项集 can，检查是否是 D 的一部分
# 即该候选 can 是否得到 transaction 的支持
    for tid in D:
        for can in Ck:
            if can.issubset(tid):
                ssCnt[can] = ssCnt.get(can, 0) + 1
    numItems = float(len(D))
    retList = []
    supportData = {}
    for key in ssCnt:
#每个项集的支持度
        support = ssCnt[key] / numItems
#将满足最小支持度的项集，加入 retList
        if support >= minSupport:
            retList.insert(0, key)
#汇总支持度数据
        supportData[key] = support
return retList, supportData
ret1,suD = scanD(loadDataSet(),createC1(loadDataSet()),0.22)
print(ret1)
print(suD)
```

输出 Out:

[frozenset({1, 2}),
frozenset({1, 3}),
frozenset({1, 4}),
frozenset({1, 5}),
frozenset({2, 3}),
frozenset({2, 4}),
frozenset({2, 5}),
frozenset({3, 4}),
frozenset({3, 5}),
frozenset({4, 5})]

这里设计了aprioriGen函数，它用于从k-1项集中生成候选k项集。20-21行调用该函数，以In[2]:中生成的单项集C1为基础，生成并输出候选2-项集。通过该表传入的参数Ck和k，可以依次在1-项集基础上生成候选2-项集，在2-项集基础上生成候选3-项集，……

一、关联规则挖掘

8. Apriori算法Python实现

```
1  #根据 minSupport 参数设定的最小支持度，返回所有满足最小支持度的项集，
2  #参数：D-List 列表，事务集
3  #       minSupport-浮点数，最小支持度阈值
4  #返回值：L-List 列表，所有满足最小支持度要求的项集
5  #          supportData-Dict 字典，项集对应的支持度
6  def apriori(D, minSupport):
7      C1=createC1(D)
8      L1, suppData = scanD(D, C1, minSupport)
9      L = [L1]
10 #最初的 L1 中的每个项集含有一个元素，
11 #新生成的项集应该含有 2 个元素，所以 k=2
12     k = 2
13     while (len(L[k-2]) > 0):
14         Ck = aprioriGen(L[k-2], k)
15         Lk, supK = scanD(D, Ck, minSupport)
16 #将新的项集的支持度数据加入原来的总支持度字典中
17         suppData.update(supK)
18 #将符合最小支持度要求的项集加入 L
19         L.append(Lk)
20 #新生成的项集中的元素个数应不断增加
21         k += 1
22 #返回所有满足条件的频繁项集的列表和所有候选项集的支持度信息
23     return L, suppData
24
25 L1,suD2=apriori(D,0.22)
26 print(L1)
27 print(suD2)
```

输出 Out:

[[frozenset({3}), frozenset({4}), frozenset({5}), frozenset({2}), frozenset({1})], [frozenset({1, 3}), frozenset({2, 3}), frozenset({2, 4}), frozenset({1, 2}), frozenset({1, 5}), frozenset({2, 5})], [frozenset({1, 2, 3}), frozenset({1, 2, 5})], []]

{frozenset({1}): 0.6666666666666666, frozenset({2}): 0.7777777777777778, frozenset({5}): 0.2222222222222222, frozenset({4}): 0.2222222222222222, frozenset({3}): 0.6666666666666666, frozenset({2, 5}): 0.2222222222222222, frozenset({1, 5}): 0.2222222222222222, frozenset({1, 2}): 0.4444444444444444, frozenset({2, 4}): 0.2222222222222222, frozenset({2, 3}): 0.4444444444444444, frozenset({1, 4}): 0.1111111111111111, frozenset({1, 3}): 0.4444444444444444, frozenset({3, 5}): 0.1111111111111111, frozenset({1, 2, 5}): 0.2222222222222222, frozenset({1, 2, 3}): 0.2222222222222222, frozenset({1, 3, 5}): 0.1111111111111111, frozenset({2, 3, 5}): 0.1111111111111111, frozenset({1, 2, 3, 5}): 0.1111111111111111}

第13行检查当前频繁k-1项集非空时，循环执行以下语句。

第14行通过k-1项集生成候选k-项集；第15行检查生成的候选k-项集是否满足最小支持度要求；

第17行记录满足最小支持度的k-项集（候选k-项集由第15行筛选而来）的支持度数据；

第19行将k项集添加到频繁项集中；第21行k自增1。

25-27行输出了所有满足最小支持度0.22要求的频繁项集列表L1和所有项集的支持度数据字典suD2。

谢谢观赏 下节课见