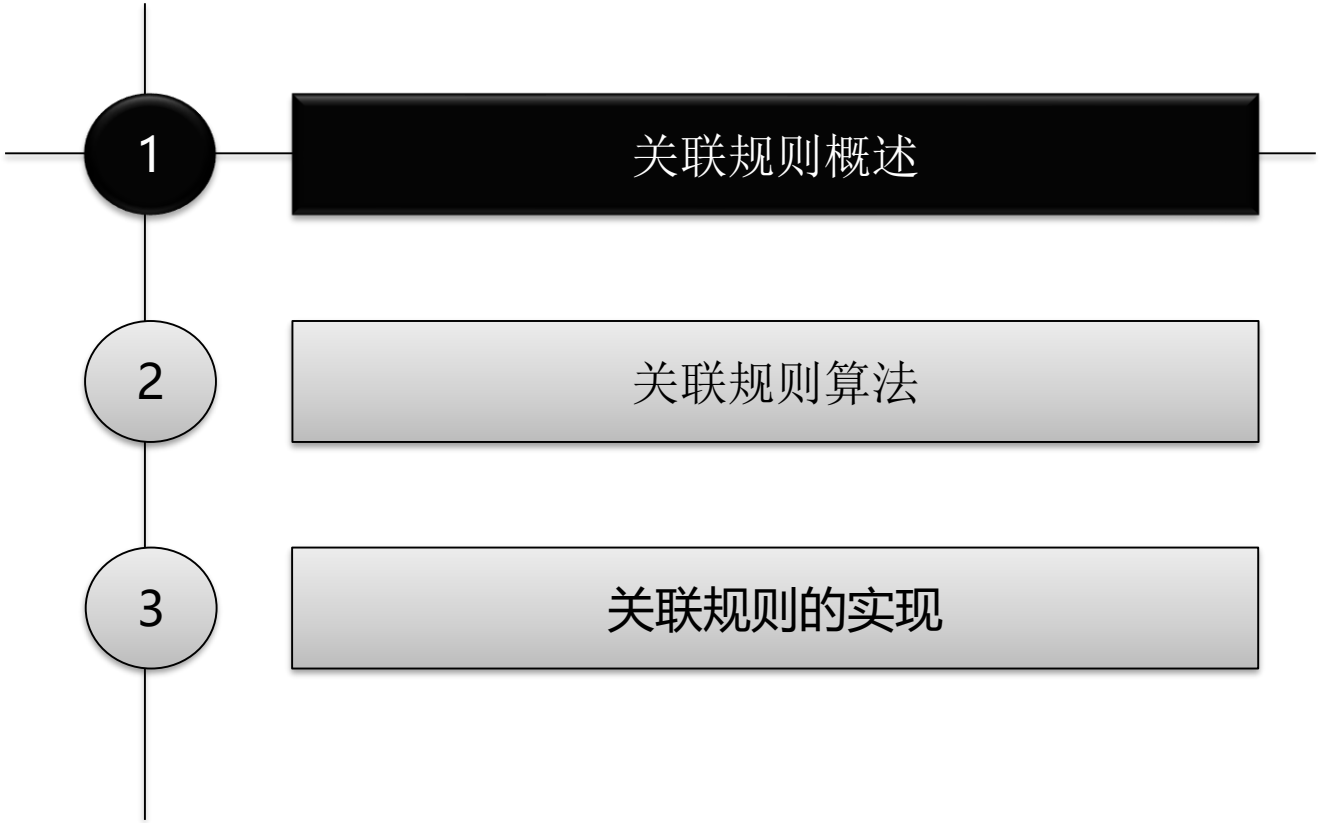
《Python与数据挖掘》

第9章 关联规则分析

2018/3/28

目录



关联规则概述

- 关联规则反映了不同事物之间的关联性，其关系通常表现为一对一或者一对多，关联规则分析是从事务数据库，关系数据库和其他信息存储中的大量数据的项集之间发现有趣的、频繁出现的模式、关联和相关性。
- 更确切的说，关联规则通过量化的数字描述物品甲的出现对物品乙的出现有多大的影响。它的模式属于描述型模式，发现关联规则的算法属于无监督学习的方法。
- 关联规则分析是数据挖掘中最活跃的研究方法之一，目的是在一个数据集中找出各项之间的关联关系，而这种关系并没有在数据中直接表示出来。

关联规则

- 常用的关联规则分析算法如下表所示：

算法名称	算法描述
Apriori	关联规则最常用也是最经典的挖掘频繁项集的算法，其核心思想是通过连接产生候选项及其支持度然后通过剪枝生成频繁项集。
Eclat算法	Eclat算法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划分为较小的子空间。
FP-Tree	针对Apriori算法的固有的多次扫面事务数据集的缺陷，提出的不产生候选频繁项集的方法。Apriori和FP-Tree都是寻找频繁项集的算法。
灰色关联法	分析和确定各因素之间的影响程度或是若干个子因素（子序列）对主因素（母序列）的贡献度而进行的一种分析方法。

- 这几种方法里，目前在Python中实现的效果较好的为Apriori算法。本章主要重点介绍Apriori算法及其在Python中的实现。

目录

Apriori算法

- Apriori算法是最经典的挖掘频繁项集的算法，第一次实现了在大数据集上可行的关联规则提取，其核心思想是通过连接产生候选项与其支持度然后通过剪枝生成频繁项集。
- 关联规则的一般形式:
- 项集A、B同时发生的概率称为关联规则的支持度（也称相对支持度）：

$$Support(A \Rightarrow B) = P(A \cap B)$$

- 项集A发生，则项集B发生的概率为关联规则的置信度：

$$Confidence(A \Rightarrow B) = P(B|A)$$

Apriori算法

b) 最小支持度和最小置信度

- ✓ 最小支持度是用户或专家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；
- ✓ 最小置信度是用户或专家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规则。

c) 项集

- ✓ 项集是项的集合。包含k个项的项集称为k项集，如集合{牛奶，麦片，糖}是一个3项集。
- ✓ 项集的出现频率是所有包含项集的事务计数，又称作绝对支持度或支持度计数。如果项集I的相对支持度满足预定义的最小支持度阈值，则I是频繁项集。频繁k项集通常记作L_k 。

Apriori算法

支持度计数

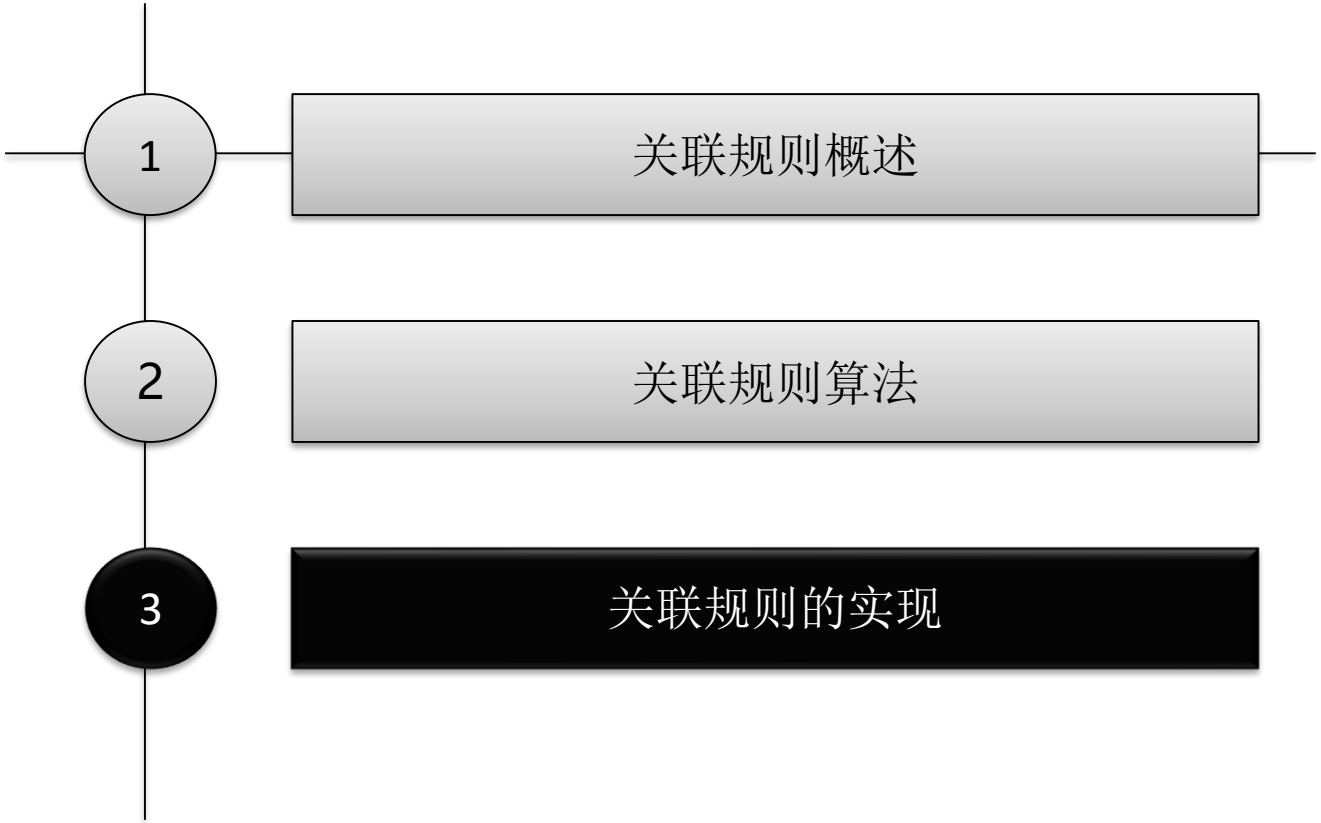
- 项集A的支持度计数是事务数据集中包含项集A的事务个数，简称为项集的频率或计数。
- 已知项集的支持度计数，则规则A $\Rightarrow B$ 的支持度和置信度很容易从所有事务计数、项集A和项集 A∪B的支持度计 数推出：

$$Support(A \Rightarrow B) = \frac{A,B\text{同时发生的事务个数}}{\text{所有事务个数}} = \frac{Support\_count(A \cap B)}{Total\_count(A)}$$

$$Confidence(A \Rightarrow B) = P(B|A) = \frac{Support(A \cap B)}{Support(A)} = \frac{Support\_count(A \cap B)}{Support\_count(A)}$$

- 也就是说，一旦得到所有事务个数，A，B和 A∪B的支持度计数，就可以导出对应的关联规则 A $\Rightarrow B$和 B $\Rightarrow A$，并可以检查该规则是否是强规则。

目录



关联规则的实现

- 下面通过餐饮企业中的例子演示Apriori在Python中的实现。
- 客户在餐厅点餐时，面对菜单中大量的菜品信息，往往无法迅速找到满意的菜品，既增加了点菜的时间，也降低了客户的就餐体验。
- 实际上，菜品的合理搭配是有规律可循的：顾客的饮食习惯、菜品的荤素和口味，有些菜品之间是相互关联的，而有些菜品之间是对立或竞争关系（负关联），这些规律都隐藏在大量的历史菜单数据中。
- 如果能够通过数据挖掘发现客户点餐的规则，就可以快速识别客户的口味，当他下了某个菜品的订单时推荐相关联的菜品，引导客户消费，提高顾客的就餐体验和餐饮企业的业绩水平。

关联规则的实现

- 数据库中部分点餐数据如下表：

序列	时间	订单号	菜品id	菜品名称
1	2014/8/21	101	18491	健康麦香包
2	2014/8/21	101	8693	香煎葱油饼
3	2014/8/21	101	8705	翡翠蒸香茜饺
4	2014/8/21	102	8842	菜心粒咸骨粥
5	2014/8/21	102	7794	养颜红枣糕
6	2014/8/21	103	8842	金丝燕麦包
7	2014/8/21	103	8693	三丝炒河粉
…	…	…	…	…

- 首先将上表中的事务数据（一种特殊类型的记录数据）整理成关联规则模型所需的数据结构，从中抽取10个点餐订单作为事务数据集，为方便起见将菜品{18491，8842，8693，7794，8705}分别简记为{a，b，c，d，e}）

关联规则的实现

- 某餐厅的业事务数据如下：

订单号	菜品id	菜品id
1	18491, 8693，8705	a，c，e
2	8842,7794	b，d
3	8842，8693	b，c
4	18491，8842，8693，7794	a，b，c，d
5	18491，8842	a，b
6	8842，8693	b，c
7	18491，8842	a，b
8	18491，8842,8693,8705	a，b，c，e
9	18491，8842,8693	a，b，c
10	18491，8693	a，c，e

- 使用Apriori函数前需要将原始数据转换为0-1矩阵，之后设置参数，data为转换好的0-1矩阵，support为最小支持度，confidence为最小置信度，ms为连接符。

关联规则实现代码

```
from __future__ import print_function
import pandas as pd
from apriori import * #导入自行编写的apriori函数
inputfile = '../data/menu_orders.xls'
outputfile = '../tmp/apriori_rules.xls' #结果文件
data = pd.read_excel(inputfile, header = None)
print(u'\n转换原始数据至0-1矩阵...')
ct = lambda x : pd.Series(1, index = x[pd.notnull(x)]) #转换0-1矩阵的过渡函数
b = map(ct, data.as_matrix()) #用map方式执行
data = pd.DataFrame(list(b)).fillna(0) #实现矩阵转换，空值用0填充
print(u'\n转换完毕。')
del b #删除中间变量b，节省内存
support = 0.2 #最小支持度
confidence = 0.5 #最小置信度
ms = '---' #连接符，默认'--'，用来区分不同元素，如A--B。需要保证原始表格中不含有该字符
find_rule(data, support, confidence, ms).to_excel(outputfile) #保存结果
```

关联规则运行结果分析

- 转换出的矩阵如下：

	a	b	c	d	e	
0	1.0	0.0	1.0	0.0	1.0	
1	0.0	1.0	0.0	1.0	0.0	
2	0.0	1.0	1.0	0.0	0.0	
3	1.0	1.0	1.0	1.0	0.0	
4	1.0	1.0	0.0	0.0	0.0	
5	0.0	1.0	1.0	0.0	0.0	
6	1.0	1.0	0.0	0.0	0.0	
7	1.0	1.0	1.0	0.0	1.0	
8	1.0	1.0	1.0	0.0	0.0	
9	1.0	0.0	1.0	0.0	1.0	

- 运行程序的结果如下：

	support	confidence
e---a	0.3	1.000000
e---c	0.3	1.000000
c---e---a	0.3	1.000000
a---e---c	0.3	1.000000
a---b	0.5	0.714286
c---a	0.5	0.714286
a---c	0.5	0.714286
c---b	0.5	0.714286
b---a	0.5	0.625000
b---c	0.5	0.625000
b---c---a	0.3	0.600000
a---c---b	0.3	0.600000
a---b---c	0.3	0.600000
a---c---e	0.3	0.600000

- 对输出结果进行解释：如关联规则“a---b　　0.5　　0.714286”这条，关联规则a---b的支持度support=0.5，置信度confidence=0.714286。对于餐饮业来说，这条规则意味着客户同时点菜品a和b的概率是50%，点了菜品a，再点菜品b的概率是71.4286%。知道了这些，就可以对顾客进行智能推荐，增加销量同时满足客户需求。

Thank You!