第11章 频繁模式挖掘算法

《人工智能算法》

清华大学出版社

2022年7月

提纲

- 频繁模式挖掘概述
- Apriori算法
- Apriori算法示例
- 总结

频繁模式挖掘概述

- **什么是频繁模式（Frequent Pattern）**

频繁地出现在数据集中的模式。例如，频繁地同时出现在新冠肺炎患者临床数据中的症状集合（发热，咳嗽，乏力），交易数据集中的商品集合（篮球、球鞋、运动裤），也称频繁项集（Frequent Itemset）。

- **频繁模式挖掘的重要性**

旨在发现数据之间隐含的关联关系，是关联规则挖掘、相关性分析、因果关系挖掘、局部周期性分析等数据挖掘任务的基础，广泛用于推荐系统、异常检测、医疗诊断等领域。

- **关联规则简介**

是形如$X \rightarrow Y$的逻辑蕴涵式，表示“通过X可推导得到Y”，X和Y为数据集中两个互不相交的事务数据集。经典的挖掘算法有Apriori、FP-Growth和Eclat等。

提纲

- 频繁模式挖掘概述
- Apriori算法
- Apriori算法示例
- 总结

Apriori算法 (1)

- **基本概念**

Transaction_ID	Items
1	A,B,C
2	A,C,D
3	A,D,E
4	B,E,F
5	B,C,D,E,F

- 最小单位信息称为项（Item）
- 项集：包含0个或多个项的集合
- 支持度（Support）：数据集中包含$X \cup Y$（出现过X和Y）的比例
- 置信度（Confidence）：规则$X \rightarrow Y$在数据集中的置信度，表示数据集中包含X的同时也包含Y的比例

- 目标是找到所有支持度和置信度不小于指定的规则：最小支持度（Minimum Support）*min_sup*，最小置信度（Minimum Confidence）*min_conf*
- 如果项集的支持度超过最小支持度阈值，称为频繁项集

Apriori算法 (2)

- **基本步骤**

Step1：寻找到所有的频繁项集，即大于或等于最小支持度阈值的项集

Step2：由频繁项集产生关联规则，且这些规则的置信度与支持度均大于或等于最小置信度阈值和最小支持度阈值

- **基本思想**

Apriori算法利用如下两个性质：1）若一个集合是频繁项集，则其所有非空子集都是频繁项集。2）若某一集合为非频繁项集，则其所有超集都是非频繁项集。

- **基本步骤**

（1）连接步：频繁项集间的并运算，将L_{k-1}中前$k-2$个项相同的$k-1$项集合并，产生候选k-项集的集合C_k。

（2）剪枝步：删除C_k中非频繁候选项集的过程，用于快速减小C_k所包含项集的数目。

Apriori算法 (3)

- **Apriori算法思想**

频繁项集的产生，需对数据集进行多步处理。

第一步，统计所有包含一个元素的项集出现的频数，并筛选出不小于最小支持度的项集；

从第二步开始循环处理，直到再没有频繁项集生成。循环过程的第k步，根据第$k-1$步生成的频繁$(k-1)$-项集产生k维侯选项集，然后搜索数据集D，得到侯选项集的支持度，并与最小支持度进行比较，从而找到频繁k-项集。

Apriori算法 (4)

用L_1表示频繁1-项集的集合
$k \leftarrow 2$；$L \leftarrow \emptyset$
While $L_{k-1} \neq \emptyset$ Do
  $C_k \leftarrow$Apriori_gen(L_{k-1}) //Apriori_gen产生候选集函数，C_k表示第k个元素的候选集
  For each $t \in D$ Do
    $count \leftarrow 0$；$C_t \leftarrow$subset(C_k, t) //C_t为D中事务t包含的所有候选集元素
    For each $c \in C_t$ Do
      $count \leftarrow count+1$
    End For
   $L_k \leftarrow \{c \in C_k | count/|D| \geq min\_sup\}$
  End For
  $L \leftarrow L \cup L_k$
  $k \leftarrow k+1$
End While
Return L

假设频繁项集的数量为N，
则Apriori的主体部分
执行$O(N^3)$次

Apriori算法 (5)

Apriori_gen(L_{k-1})　//产生候选集函数

$C_k \leftarrow \emptyset$

For each $p \in L_{k-1}$ Do

　For each $q \in L_{k-1}$ Do

　　If $p[1]=q[1] \wedge \cdots \wedge q[k-2]=q[k-2] \wedge p[k-1]<q[k-1]$ Then

　　$c \leftarrow p \cup q$　//把q的第$k-1$个元素连接到p

　　If Has_infrequent_subset(c, L_{k-1}) Then　//判断c是否为候选集

　　$C_k \leftarrow C_k \backslash \{c\}$

　　Else

　　$C_k \leftarrow C_k \cup \{c\}$

　　End If

　End For

End For

Return C_k

连接步

时间复杂度：
$O(N^2)$

Apriori算法 (4)

Has_infrequent_subset(c, L_{k-1}) //判断c是否为候选集的函数
For each $s \in c^{(k-1)}$ Do //用$c^{(k-1)}$表示c的($k-1$)-子集的集合
  If $s \notin L_{k-1}$ Then
    Return True
  Else
    Return False
  End If
End For

剪枝步

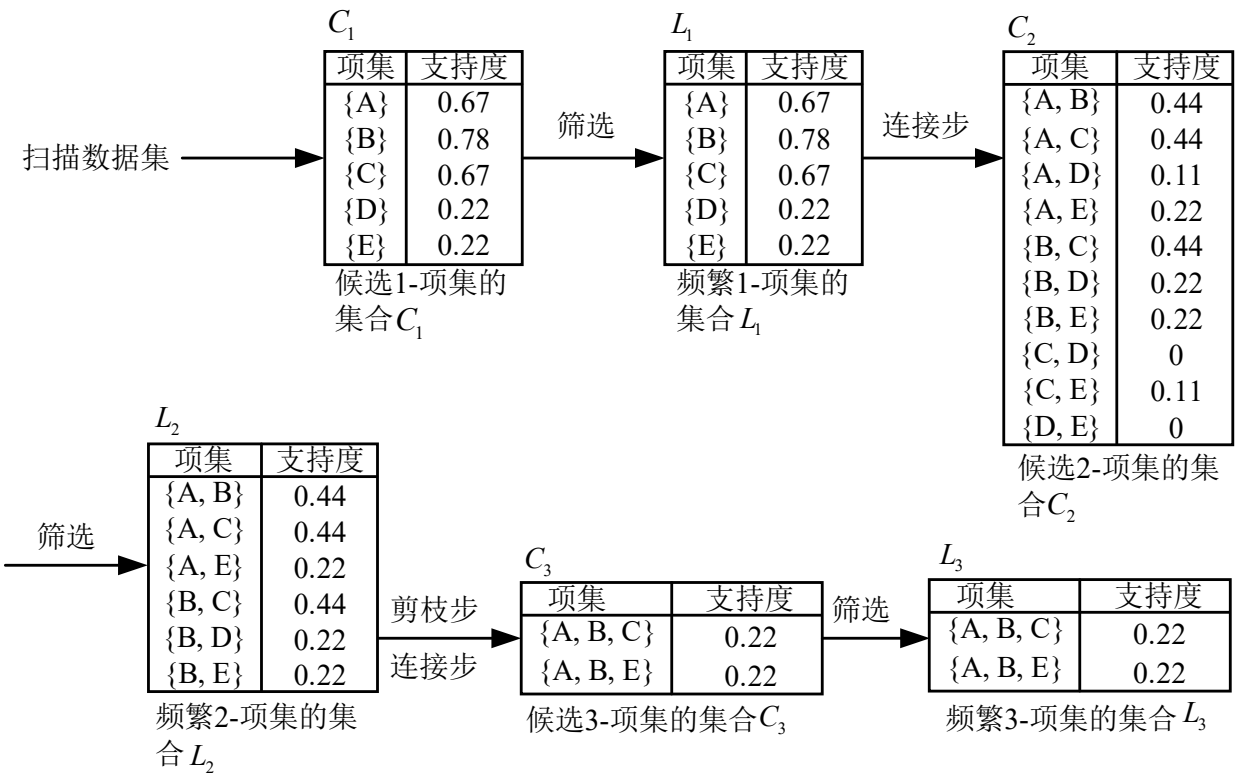
函数Has_infrequent_subset(c, L_{k-1})的时间复杂度为$O(M)$。
因此，Apriori算法的时间复杂度为$O(N^5 \times M)$。

提纲

- 频繁模式挖掘概述
- Apriori算法
- Apriori算法示例
- 总结

Apriori算法示例 (1)

TID	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
项ID	A B E	B D	B C	A B D	A C	B C	A C	A B C E	A B C



Apriori算法示例 (2)

- 首先扫描所有事务，对每个项的出现次数计数，产生1-项集的集合C_1，通过比较候选项集的支持度与最小支持度，确定频繁1-项集L_1。
- 产生L_1中所有可能的组合，并对每个组合进行拆分，保证频繁项集的所有非空子集也是频繁的。计算C_1中项集的支持度，得到频繁2-项集L_2，比较候选集C_2与最小支持度，产生候选2-项集C_2，产生候选3-项集的集合C_3。
- 由于频繁项集的所有子集必须是频繁的，可确定后4个候选项集不可能是频繁的，因此从C_3中删除。扫描D中事务，由具有最小支持度的C_3中的候选3-项集组成L_3。
- 再利用L_3产生候选4-项集的集合C_4，可知C_4=∅，即找出所有频繁项集。
- 基于挖掘得到的所有频繁项集，生成关联规则：

  （1）对于每个频繁项集l，生成l的所有的非空子集；

  （2）对于l的每一个非空子集s，若$\frac{Sum(l)}{Sum(s)} \geq min_conf$，则输出规则“$s \rightarrow (l-s)$”。

Apriori算法示例 (3)

基于挖掘得到的频繁项集L={A, B, E}，生成关联规则：

（1）根据频繁项集L={A, B, E}，得到非空子集为{A}, {B}, {E}, {A, B}, {A, E}, {B, E}；

（2）对每一个非空子集，计算频繁项集L在数据集D中出现的次数与非空子集出现次数的比值，若min_conf=0.7，则规则E→A∧B、A∧E→B和B∧E→A为强规则。

基于频繁项集产生关联规则：

非空子集	置信度	关联规则
{A}	2/6=0.33	A → B ∧ E
{B}	2/7=0.29	B → A ∧ E
{E}	2/2=1	E → A ∧ B
{A, B}	**2/4=0.5**	A ∧ B → E
{A, E}	2/2=1	A ∧ E → B
{B, E}	2/2=1	B ∧ E → A

提纲

- 频繁模式挖掘概述
- Apriori算法
- Apriori算法示例
- 总结

总结

Apriori算法的优缺点：

- 优点：简单、易理解、数据要求低，对稀疏的、短的频繁模式挖掘具有较高的效率，且扩展性好、可并行计算。
- 缺点：在每一步产生侯选项集时循环产生的组合过多，没有排除不应参与组合的元素；每次计算项集的支持度时，都对数据集D中的全部记录进行一遍扫描比较，对于大型的数据集，这种扫描比较将大大增加计算时间开销，且这种代价随着数据集中记录数的增加呈现出几何级数增加。

结语

谢谢！