1.Apriori算法简介

Apriori算法是常用于挖掘出数据关联规则的算法，能够发现事物数据库中频繁出现的数据集，这些联系构成的规则可帮助用户找出某些行为特征，以便进行企业决策。例如，某食品商店希望发现顾客的购买行为，通过购物篮分析得到**大部分顾客会在一次购物中同时购买面包和牛奶**，那么该商店便可以通过降价促销面包的同时提高面包和牛奶的销量。了解Apriori算法推导之前，我们先介绍一些基本概念。

- **事务数据库：**设$I=\{i_1,i_2,\ldots,i_m\}$是一个全局项的集合，事物数据库$D=\{t_1,t_2,..,t_n\}$是一个事务的集合，每个事务$t_i(1\leq i\leq n)$都对应I上的一个子集，例如$t_1=\{i_1,i_3,i_7\}$。
- **关联规则：**关联规则表示项之间的关系，是形如$X\rightarrow Y$的蕴含表达式，其中X和Y是不相交的项集，X称为规则的前件，Y称为规则的后件。例如$\{cereal,milk\}\rightarrow\{fruit\}$关联规则表示购买谷类食品和牛奶的人也会购买水果。通常关联规则的强度可以用支持度和置信度来度量。
- **支持度：**支持度表示关联数据在数据集中出现的次数或所占的比重。

$$support(X\rightarrow Y)=P(X\cup Y)=\frac{|X\cup Y|}{|D|}$$

- **置信度：**置信度表示Y数据出现后，X数据出现的可能性，也可以说是数据的条件概率。

$$confidence(X\Leftarrow Y)=P(X|Y)=\frac{P(XY)}{P(Y)}$$

- **提升度：**提升度体现X和Y之间的关联关系，提升度大于1表示X和Y之间具有强关联关系，提升度小于等于1表示X和Y之间无有效的强关联关系。

$$lift(X\Leftarrow Y)=\frac{confidence(X\Leftarrow Y)}{P(X)}=\frac{P(XY)}{P(X)P(Y)}$$

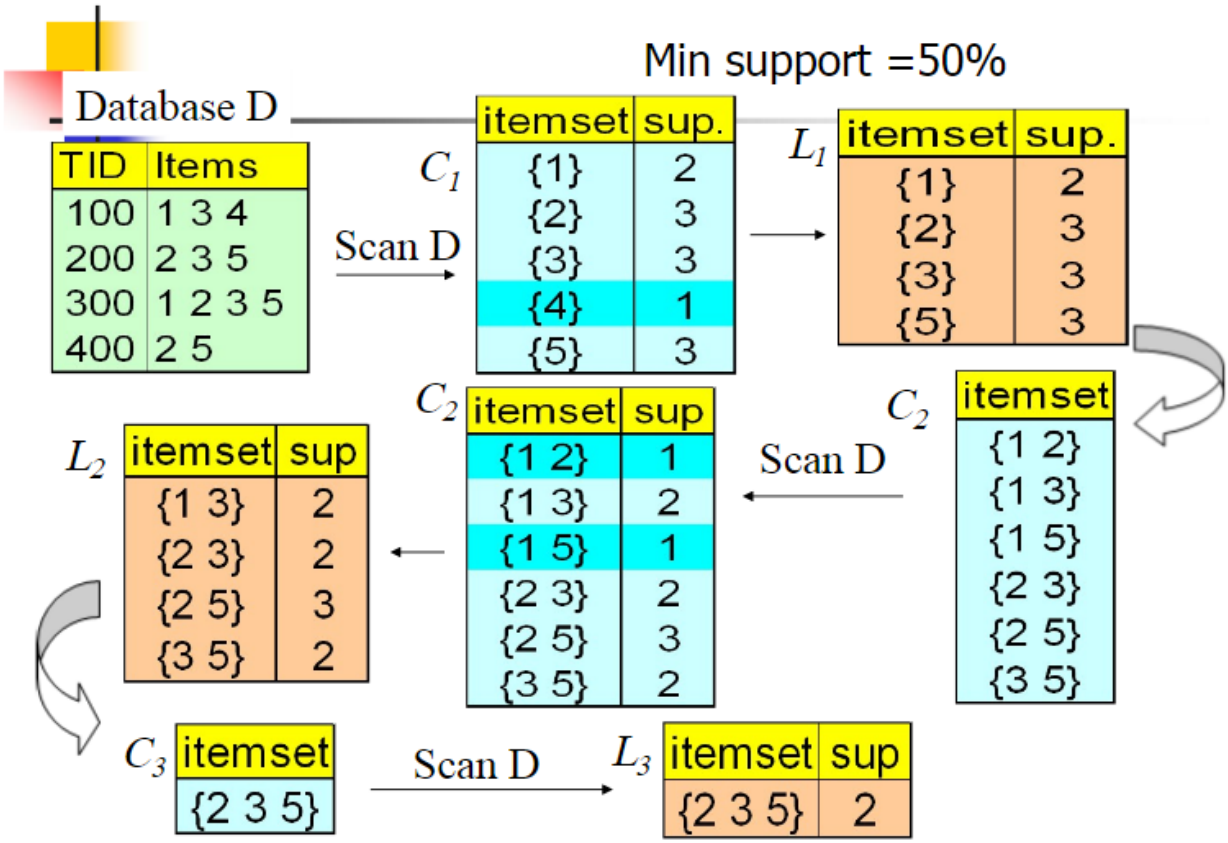
- **强关联规则：**满足最小支持度和最小置信度的关联规则。

关联规则的挖掘目标是**找出所有的频繁项集**和**根据频繁项集产生强关联规则**。对于Apriori算法来说，其目标是找出所有的频繁项集，因此对于数据集合中的频繁数据集，我们需要自定义评估标准来找出频繁项集，常用的评估标准就是用上述介绍的支持度。

2.Apriori算法原理

Apriori算法是经典生成关联规则的频繁项集挖掘算法，其目标是**找到最多的K项频繁集**。那么什么是最多的K项频繁集呢？例如当我们找到符合支持度的频繁集AB和ABE，我们会选择3项频繁集ABE。下面我们介绍Apriori算法选择频繁K项集过程。

Apriori算法采用迭代的方法，先搜索出候选1项集以及对应的支持度，剪枝去掉低于支持度的候选1项集，得到频繁1项集。然后对剩下的频繁1项集进行连接，得到候选2项集，筛选去掉低于支持度的候选2项集，得到频繁2项集。如此迭代下去，直到无法找到频繁k+1集为止，对应的频繁k项集的集合便是算法的输出结果。我们可以通过下面例子来看到具体迭代过程。



数据集包含4条记录{'134','235','1235','25'}，我们利用Apriori算法来寻找频繁k项集，最小支持度设置为50%。首先生成候选1项集，共包含五个数据{'1','2','3','4','5'}，计算5个数据的支持度，然后对低于支持度的数据进行剪枝。其中数据{4}支持度为25%，低于最小支持度，进行剪枝处理，最终频繁1项集为{'1','2','3','5'}。根据频繁1项集连接得到候选2项集{'12','13','15','23','25','35'}，其中数据{'12','15'}低于最低支持度，进行剪枝处理，得到频繁2项集为{'13','23','25','35'}。如此迭代下去，最终能够得到频繁3项集{'235'}，由于数据无法再进行连接，算法至此结束。

3.Apriori算法流程

从Apriori算法原理中我们能够总结如下算法流程，其中输入数据为数据集合D和最小支持度α，输出数据为最大的频繁k项集。

- 扫描数据集，得到所有出现过的数据，作为候选1项集。
- 挖掘频繁k项集。
    - 扫描计算候选k项集的支持度。
    - 剪枝去掉候选k项集中支持度低于最小支持度α的数据集，得到频繁k项集。如果频繁k项集为空，则返回频繁k-1项集的集合作为算法结果，算法结束。如果得到的频繁k项集只有一项，则直接返回频繁k项集的集合作为算法结果，算法结束。
    - 基于频繁k项集，连接生成候选k+1项集。
- 利用步骤2，迭代得到k=k+1项集结果。

4.Apriori算法优缺点

4.1优点

- 适合稀疏数据集。
- 算法原理简单，易实现。

- 适合事务数据库的关联规则挖掘。

4.2缺点

- 可能产生庞大的候选集。
- 算法需多次遍历数据集，算法效率低，耗时。

5.推广

更多内容请关注公众号**谓之小一**，若有疑问可在公众号后台提问，随时回答，欢迎关注，内容转载请注明出处。

参考

> 刘建平Pinard-Apriori算法原理总结