使用垂直数据格式挖掘频繁项集

陈 伟
(淮南联合大学 计算机系,安徽 淮南 232038)

摘 要:关联规则是数据挖掘的主要技术之一,它是描述数据库中一组数据项之间的某种潜在关系的规则。关联规则挖掘算法——Apriori 算法,主要过程是对频繁项集的挖掘,而在对频繁项集的挖掘中首先要生成候选频繁项集,然后再从候选集中确定出满足最小支持度计数的频繁项集,这会耗费大量的 CPU 开销。使用垂直数据格式挖掘频繁项集可避免候选项目集的求解。

关键词:关联规则;Apriori 算法;频繁项集;垂直数据格式
中图分类号:TP312 文献标识码:A 文章编号:1674-7720(2011)18-0006-02

Uses the vertical data format to unearth the frequent itemsets

Chen Wei
(Huainan Union University, Huainan Anhui 232038, China)

Abstract: Association Rule is one of the key technologies of data mining. It describes the potential relations of a group of data items in database. Connection rule excavation algorithm - Apriori algorithm, The main process is to extract the frequent itemsets, however in the frequent item set's excavation must be first produce the candidate frequent itemsets, Then determined again from the candidate frequent itemsets satisfies the smallest support counting the frequent itemsets, This will consume the massive cpu expenses. Used the vertical data format excavation frequent itemsets to avoid the candidate itemsets solution.

Key words: association rule; Apriori algorithm; frequent itemsets; vertical data format

通常,关联规则挖掘是指从一个大型的数据集中发现有趣的关联或相关关系,即从数据集中识别出频繁出现的属性值集(Sets of Attribute-Values),也称为频繁项集(Frequent Itemsets,简称频繁集),然后再利用这些频繁集创建描述关联规则的过程。

1 关联规则挖掘算法

关联规则挖掘算法——Apriori 算法是使用候选项集找频繁项集的过程。

Apriori 算法通过对数据库 D 的多趟扫描来发现所有的频繁项目集。在第一趟扫描数据库时,对项集 I 中的每一个数据项计算其支持度,确定出满足最小支持度的频繁 1 项集的集合 L_1,然后,L_1 用于找频繁 2 项集的集合 L_2,如此下去……在后续的第 k 次扫描中,首先以 $k-1$ 次扫描中所发现的含 $k-1$ 个元素的频繁项集的集合 L_{k-1} 为基础,生成所有新的候选项目集 C_K(Candidate Itemsets),即潜在的频繁项目集,然后扫描数据库 D,计算这些候选项目集的支持度,最后从候选集 C_K 中确定出满足最小支持度的频繁 k 项集的集合 L_k,并将 L_k 作为下一次扫描的基础。重复上述过程直到不再发现新的频繁项目集[1]。

2 关联规则算法的改进

从 Apriori 算法中由 k 频繁项集生成 $k+1$ 频繁项集时,首先生成候选项目集 C_{k+1},该函数不仅要对 k 项集的所有符合 Apriori 算法条件的数据进行交集,并且要判断候选项目集的所有子集是否在 k 频繁项集中。该函数生成的许多候选项目集并不是要找的频繁项集,但在扫描数据库时,要记录下这些数据的出现次数,这会耗费大量的 CPU 开销。如果 D 中的事务数很大,k 频繁项集中项数很多,则候选项目集的元素个数就会很大,例如 2 000 个频繁 1 项集,将产生 2 000×999/2=999 000 个候选 2 项集。如此巨大数量的候选项目集,对它进行出现次数的统计时开销非常大,这也是整个算法性能优劣的关键所在。

(1)使用垂直数据格式挖掘频繁项集

Apriori 算法是从 TID 项集格式(即{TID:itemset})的事务集挖掘频繁模式,其中 TID 是事务标识符,而 item-

set 是事务 TID 中购买的商品集。这种数据格式称作水平数据格式。另外,数据也可以用项-TID 集格式(即{item: TID_set})表示,其中 item 是项的名称,而 TID_set 是包含 item 事务标识符的集合。这种格式称作垂直数据格式。下面使用垂直数据格式进行有效的挖掘频繁项集。

首先,通过扫描一次数据库 D,在求频繁 1 项集的同时,把数据由水平格式转化为垂直格式,即记录下每个项集在事务数据库中出现时该条数据在数据库 D 中的 TID 号,则项集的支持度计数直接是项集的 TID 集的长度。从 $k=2$ 开始,根据 Apriori 性质中相交条件的项集进行 Apriroi 连接运算,使用频繁 k 项集来构造候选 $k+1$ 项集。通过取频繁 k 项集的 TID 集的交计算对应的 $k+1$ 项集的 TID 集。如果该 TID 集的长度大于最小支持度计数,则该记录为频繁项集。重复该过程,每次 k 值增加 1,直到不能再找到频繁项集或候选项集。

(2)挖掘实例

设事务数据库 D,如表 1。

表 1 事务数据库

TID	Item
1	1,3,4
2	2,3,5
3	1,2,3,5
4	2,5

假定要求最小支持度计数为 2,扫描一次该数据集可以把它转换成表 2 所示的垂直数据格式。

表 2 频繁 1 项集

Item	TID 集	是否频繁 1 项集
I1	{T1,T3}	是
I2	{T2,T3,T4}	是
I3	{T1,T2,T3}	是
I4	{T1}	否
I5	{T2,T3,T4}	是

通过取每对频繁单个项的 TID 集的交,可以在该数据集上进行挖掘。由于表 2 中有 4 个项是频繁的,总共进行 6 次交运算,结果有 6 个非空 2 项集,如表 3 所示。

表 3 频繁 2 项集

Item	TID 集	是否频繁 2 项集
{I1,I2}	{T3}	否
{I1,I3}	{T1,T3}	是
{I1,I5}	{T3}	否
{I2,I3}	{T2,T3}	是
{I2,I5}	{T2,T3,T4}	是
{I3,I5}	{T2,T3}	是

根据 Apriori 性质,对得到的频繁 2 项集求交集,只进行 1 次交集,所以符合条件的频繁三项集为{I2,I3,I5}。如表 4 所示。

表 4 频繁 3 项集

Item	TID 集	是否频繁 3 项集
{I2,I3,I5}	{T2,T3}	是

通过挖掘可以验证最终得到的频繁项集和用 Apriori 算法得到的结果是相同的[2-3]。

(3)算法的描述

输入:事务数据库 D;

最小支持度计数阈值 min_sup。

输出:D 中的频繁项集 L。

```
L1=Apriori_genl(D,min_sup);
For(k=2;Lk-1≠Φ;k++) do begin
Lk=Apriori_genK(Lk-1,min_sup)
End
Return L=UkLk
```

输入:事务数据库 D;最小支持度计数 min_sup。

输出:所有的频繁 1 项集。

```
Procedure Apriori_genl(D,min_sup)
L1= find_frequent_1-itemsets(D);
                //定义 L1 用来保存所有找到的频繁 1 项集
For all transactions t∈D
do begin
 For(i=O;i<t.item[];i++) Do begin
  If(t.item[i]∈L1)
 {L1[j].Tid+=t.TID;    //把找到的项目所对应的 TID
                       号加入到 L1 表中
  L1[j].count++;       //对找到的 TID 号计数
  }
 End
End
For all L1[j].count do begin
If(L1[j].count<min_sup)
 delete(L1[j]);
End
```

输入:频繁 k-1 项集。最小支持度计数 min_sup。

输出:所有的频繁 k 项集。

```
Procedure Apriori_genK(Lk-1,min_sup)
Lk=Null;//定义一个链表
For each itemset L1∈Lk-1
For each itemset L2∈Lk-1
    if((L1[1]=L2[1])∧(L1[2]=L2[2]) ∧…∧(L1[k-2]=
    L2[k-2]) ∧(L1[k-1]< L2[k-1]))
        Do begin
        Lk[i].Tid= L 1.Tid L 2.Tid
        If(Lk[i].Tid.length>=min_sup)
            Lk.add(item);
    End
Return Lk[4-6]
```

3 改进算法的分析

理论上,改进算法应快于算法 Apriori,原因如下:

(1)该算法不用求候选频繁项集,从而省去了项集进行连接步以后对该项集所有子集的判断,这会节省大量

的 CPU 开销，尤其是当数据库 D 中的数据比较多，同时候选项集比较多时，例如频繁 1 项集 2 000 个，根据 Apriori 性质，则候选频繁 1 项集要有2 000×999/2=999 000 个，但其中也许只有少数是要找的频繁 2 项集。Apriori 算法要求其中所有 999 000 个在数据库中的频繁度。

(2) 除了由频繁 k 项集产生候选 $k+1$ 项集时利用 Apriori 性质之外，这种方法的另一优点是不需要扫描数据库来确定 $k+1$ 项集($k \geq 1$)的支持度。这是因为每个 k 项集的 TID 集携带了计算该支持度所需的完整信息。

该算法的缺点是 TID 集可能很长，长集合不仅需要大量空间，而且求交运算需要大量计算时间。

参考文献

[1] 韩家炜. 数据挖掘概念与技术[M].北京：机械工业出版社，2000.

[2] 陆楠.关联规则的挖掘及其算法的研究[D]. 长春：吉林大学，2007.

[3] 罗可，张学茂. 一种高效的频集挖掘算法[J]. 长沙理工大学学报(自然科学版)，2006，3(3)：84-90.

[4] 汪光一. 关联规则挖掘算法的研究[D]. 北京：北京交通大学，2007.

[5] 高峰，谢剑英. 发现关联规则的增量式更新算法[J]. 计算机工程，2000(12)：49-50.

[6] KANTARDZIC M(美).数据挖掘-概念、模型、方法和算法[M]. 北京：清华大学出版社，2003.

(收稿日期：2011-04-20)

作者简介：

陈伟，女，1973 年生，讲师，硕士，主要研究方向：数据挖掘。