

订阅DeepL Pro以编辑此演示文稿。  
访问www.DeepL.com/Pro，了解更多信息。

*挖掘关联规则的快速算法Rakesh AgrawalRamakrishnan Srikant\*。*

# *IBM Almaden研究中心*

*哈利路650号。圣何塞。 CA95120*

*摘要*

*我们考虑在一个大型的销售交易数据库中发现项目之间的关联规则问题。我们提出了两种解决这个问题的新算法，它们与已知的算法有根本的不同。经验评估表明，这些算法的性能优于已知的算法，对于小问题来说，其系数为3，对于大问题来说，其系数超过了一个数量级。我们还展示了如何将两种拟议算法的最佳特征结合到一个混合算法中，称为AprioriHybrid。扩展实验表明，AprioriHybrid随着交易数量的增加而线性地扩展。AprioriHybrid在交易规模和数据库中的项目数量方面也具有出色的扩展特性。*

# *1简介*

*条形码技术的进步使*零售机构*有可能*收集和储存大量的销售数据，即所谓的篮子数据。这些数据中的一条记录通常包括交易日期和交易中购买的物品。成功的组织将此类数据库视为营销基础设施的重要部分。他们有兴趣建立信息驱动的营销流程，由数据库技术管理，使营销人员能够开发和实施定制的营销方案和战略6] 。

4]中介绍了*在篮子*数据*上挖掘关联规则的问题。* 这样一个规则的例子可能是：98%的客户购买了



*\* 来自威斯康星大学麦迪逊分校计算机科学系的访问学者。*

*允许无偿复制本材料的全部或部分内容*

*只要不为直接的商业利益而制作或分发副本，出现VLDB版权声明和出版物的标题及其日期，并通知复制是由Very Large Data Base Endowment许可的，就可以获得授权。以其他方式复制，或重新发表，需要付费和/或获得捐赠基金的特别许可。*

*第20届VLDB会议论文集，智利 圣地亚哥，1994年*

*轮胎和汽车配件也得到了汽车服务*。找到所有这些规则对交叉营销和附加邮件应用很有价值。其他应用包括目录设计、附加销售、商店布局和基于购买模式的客户细分。这些应用所涉及的数据库是非常大的。因此，当务之急是要有快速的算法来完成这项任务。

*以下是对问题的正式陈述*

*4]: 让I = fil, i2, ..., img是一组字词，*称为项目。让D是一个交易集，其中每个交易T是一个项目集，使得T C I.与每个交易相关的是一个唯一的标识符，称为其TID。如果X C T，我们就说一个交易T包含X，即I中的一些项目的集合。关联规则是一个形式为X=⇒Y的暗示，其中X C I，Y C I，并且X n Y = *0。*

*规则X=⇒Y在交易集D中成立，其中*

*如果D中包含*X的*交易的c%*也包含Y，则*信心为c*。 如果D中s%的交易包含X U Y，则规则X =⇒ Y在交易集D中具有支持度s。我们的规则比4]中的规则更通用，因为我们允许一个结果有一个以上的项目。

*给定一组事务D，*最小化关联规则*的问题*是生成所有支持度和置信度分别大于用户指定的最小支持度（称为minsup）和最小置信度（称为minconf）的关联规则。例如，D可以是一个数据文件，一个关系表，或者一个关系表达的结果。

4]中预先提出了*一种寻找所有关联规则的算法，*此后被称为AIS算法。13]中提出了另一种用于这一任务的算法，称为SETM算法。在本文中，我们提出了两种新的算法，Apriori和AprioriTid，它们与这些算法有着本质的区别。我们提出的实验结果表明

*所提出的算法总是优于*早期的算法。性能上的差距随着问题的大小而增加，对于小问题来说，差距为三倍，对于大问题来说，差距超过一个数量级。然后，我们讨论了如何将Apriori和Apriori-Tid的最佳特征结合到一个混合算法中，称为AprioriHybrid。实验表明，Apriori-Hybrid具有良好的扩展特性，为在非常大的数据库中挖掘关联规则提供了可行性。

*寻找关联规则的问题属于*

*属于数据库挖掘的范畴3] 12]，也称为*数据库中的知识发现21]。相关的，但不直接适用的工作包括分类规则的归纳8] 。11]22]、因果规则的发现19]、逻辑定义的学习18]、函数与数据的拟合15]、以及聚类9] 。10].机器学习文献中最接近的工作是20]中提出的KID3算法。如果用于寻找所有的关联规则，这个算法将在数据上进行的次数与前述项目组合的数量一样多，而这是指数级的大。数据库文献中的相关工作是关于从数据中推断出功能依赖的工作16]。功能性依赖是需要严格满足的规则。因此，在确定了一个依赖关系X---A之后，16]中的算法认为任何其他形式的X+Y---A的依赖关系都是多余的，不会产生。我们考虑的关联规则在本质上是概率性的。规则X --- A的存在不一定意味着X + Y --- A也成立，因为后者可能没有最小支持。同样，规则X --- Y和Y --- Z的存在不一定意味着X --- Z成立，因为后者可能没有最小支持度。

*在量化*规则的 *"有用性* "或 "有趣性 "*方面，已经有了一些工作成果*20]。什么是有用的或有趣的，往往取决于应用。例如，在7]中已经阐述了人类在循环中的需要，并提供工具以允许人类对规则发现过程进行指导。14].我们在本文中不讨论这些问题，只是指出这些都是可能使用我们的算法作为发现过程的引擎的规则发现系统的必要特征。

*1.1问题分解和论文组织*

*发现所有关联规则的问题可以被分解*为两个子问题4]。

*1.找到所有项目集（项目集），这些项目集的交易*支持度高于最小支持度。支持率

*一个项目集的最小端口是*包含该项目集的*交易的数量*。具有最小超级端口的项目集被称为大项目集，而所有其他的小项目集。在第2节，我们给出了解决这个问题的新算法，Apriori和AprioriTid。

*2.使用大项目集来生成所需的规则。*下面是这个任务的一个简单的算法。对于每一个大项集l，找到l的所有非空子集。对于每一个这样的子集a，如果support(l)和support(a)的比率至少是minconf，则输出一个形式为a =⇒(l - a)的规则。我们需要考虑l的所有子集来生成具有多个结果的规则。由于篇幅有限，我们不进一步讨论这个子问题，但请读者参考5]中的快速算法。

*在第3节中，我们展示了*拟议的Apriori和AprioriTid算法与AIS 4]和SETM 13]算法的*相对性能*。为了使本文自成一体，我们在本节中包括了对AIS和SETM算法的概述。我们还描述了如何将Apriori和AprioriTid算法组合成一个混合算法，即AprioriHybrid，并展示了这个算法的扩展特性。最后，我们在*第四节*中指出了一些相关的开放问题*。*

# *2. 发现大项目集*

*发现大项目集的算法会对*数据进行*多次*处理。在第一遍中，我们计算各个项目的支持度，并确定其中哪些是大项目，即具有最小支持度。在随后的每一次传递中，我们从上一次传递中发现的大项目集的种子集开始。我们使用这个种子集来生成新的潜在的大项目集，称为候选项目集，并在通过数据时计算这些候选项目集的实际支持率。在传递结束时，我们确定哪些候选项集是真正的大项集，它们将成为下一次传递的种子。这个过程一直持续到没有新的大项目集被发现为止。

*我们提出的Apriori和AprioriTid算法*与AIS 4]和SETM 13]算法的根本区别在于，在一个通道中计算哪些候选项集以及这些候选项集的生成方式。在AIS和SETM算法中，候选项目集是在读取数据的过程中即时生成的。具体来说，在读取一个交易后，要确定哪些在前一次传递中发现的大项目集存在于交易中。新的候选项目集是通过用交易中的其他项目扩展这些大项目集而产生的。然而，正如我们将看到的，其缺点是

*是，这导致不必要地生成和*计算太多的候选项集，而这些候选项集最终都是小的。

*Apriori和AprioriTid算法*通过只使用前一次发现的大项目集来*生成*要计算的候选项目集--不考虑数据库中的事务。基本的直觉是，大项目集的任何子集一定是大的。因此，有k个项目的候选项目集可以通过连接有k-1个项目的大项目集，并删除那些包含不大的子集的项目来生成。这个过程的结果是生成一个数量少得多的候选项集。

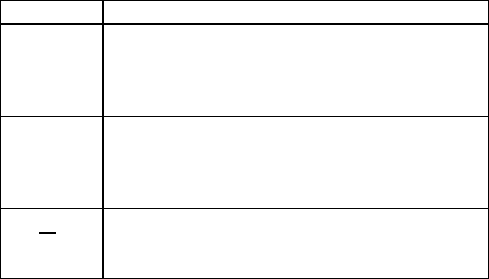
*AprioriTid算法有一个额外的*特点，就是在第一遍之后，数据库根本不用来计算候选项集的支持度。相反，前一阶段使用的候选项集的编码被用于这一目的。在后面的过程中，这个编码的大小可以变得比数据库小得多，从而节省了大量的阅读工作。我们将在描述算法时更详细地解释这些要点。

*符号 我们假设每个交易中的项目*都是按照它们的词法顺序排序的。将这些算法改编为数据库D保持正常化的情况是很简单的，每个数据库记录是一个< TID, item>对，其中TID是相应交易的标识。

*我们把*一个*项目集中的项目数量称为大小，*并把大小为k的项目集称为k-项目集。项目集中的项目是按词汇表的顺序保存的。我们使用的符号是c 1]- • c 2]- • ...- c k]来表示一个由项目c 1], c 2], ... c k]组成的k-项目集，其中c 1] < c 2] < ... < c k]。 如果c = X - Y，并且Y是一个m项集，我们也称Y为X的m扩展。与每个项集相关的是一个计数字段，用来存储这个项集的支持度。当项目集第一次被创建时，计数字段被初始化为零。

*我们在表1中总结了算法中使用的符号*。AprioriTid使用的是集合Ck，并且将

*表1：记号*



*k-itemset*

*Lk*

*Ck*

*Ck*

*一个有k个项目的项目集。*

*大K项集的集合*

*(有最低限度支持的人)。*

*这个集合的每个成员都有两个字段。*

*i）项目集和ii）支持数。*

*候选K项集的集合（可能是大项集）。*

*这个集合的每个成员都有两个字段。*

*i）项目集和ii）支持数。*

*当生成交易的TID与候选项保持关联时，候选K项集的集合。*

*2.1.2节描述了*用于此目的的*子集*函数。关于缓冲区管理的讨论见5]。

*1) L1 = flarge 1-itemsetsg;*

*2) for ( k = 2; Lk-1 = 0; k++ ) do begin*

*3) Ck = apriori-gen(Lk-1); 新候选人*

*4) forall transactions t E D do begin*

*5) Ct = subset(Ck , t); t中包含的候选者*

*6) forall candidates c E Ct do*

*7) c. count++。*

*8) 结束*

*9)Lk = fc E Ck j c. count 2: minsupg*

*10）结束*

*11)答案 = Sk Lk ;*

*图1：算法Apriori*

*2.1.1 Apriori候选者的生成*

*apriori-gen函数的参数是Lk\_l，即*所有大（k-1）-项目集的集合。它返回一个所有大的k-项目集的超集。l 首先，在连接步骤中，我们将Lk\_l与Lk\_l连接起来。

*插入Ck中*

*选择p.项目 , p.项目 , ..., p.项目 , q.项目*

*我们在描述这个算法时将进一步讨论。*

*12*

*从Lk-1 p，Lk-1 q*

*k-1*

*k-1*

*2.1算法Apriori*

*图1给出了Apriori算法。*该算法的*第一遍*只是计算项目的出现次数，以确定大的1-项目集。接下来的一个环节，比如说第k个环节，由两个阶段组成。首先，在第(k-1)遍中发现的大项目集Lk\_l被用于

*其中p.item1 = q.item1 ， . . ., p.itemk-2 = q.itemk-2 , p.itemk-1 < q.itemk-1 。*

*接下来，在修剪步骤中，我们删除所有项目集c E Ck*，使c的一些（k-1）子集不在Lk\_l中。



*1 与我们的工作同时，17号文件中提出了以下两步候选人生成程序]*

*三*

*I*

*产生候选项集Ck，使用priori-*

*2.1.1节中描述的基因函数。 接下来，*

*Ck = fX u X jX, X E L I*

*k-1*

*，jX n X j = k - 2g*

*扫描数据库，并计算*Ck*中候选人的支持率*。为了快速计数，我们需要有效地确定Ck中的候选者包含在一个

*Ck = fX E Ck jX 包含Lk-1 g的k个成员*

*这两个步骤分别类似于我们的连接和修剪步骤。然而，一般来说，步骤1会产生一个由我们的连接步骤产生的候选人的超集。*

*forall itemsets c E Ck do*

*后来，当计算和保持的成本在*

*forall (k-1)-subsets of c do*

*k l*

*存储器附加C/*

*- Ck l*

*候选人变成*

*如果(s E Lk-1)那么*

*从Ck中删除c。*

*例子 让L3是ff1 2 3g, f1 2 4g, f1 3 4g, f1* 3 5g, f2 3 4gg。在连接步骤之后，C4将是ff1 2 3 4g, f1 3 4 5g g。剪枝步骤将删除项目集f1 3 4 5g，因为项目集f1 4 5g不在L3中。然后，我们将只剩下C4中的f1 2 3 4g。

将*这种候选生成方式与*AIS和SETM算法中使用的*方式进行对比*。 在这些算法的通道k中，读取一个数据库事务t，并确定Lk\_l中的哪些大项集出现在t中。然后，这些大项集中的每一个l都被扩展为t中出现的所有大项，并且在词法排序中比l中的任何项都要晚。

*3 4 5g. 在第四道工序中，AIS和SETM将*

通过扩展大项集f1 2 3g，*产生两个候选项集，即f1 2 3 4g和f1 2 3 5g*。同样，通过扩展L3中的其他大项集，还将产生另外三个候选项集，从而导致在第四遍中总共有5个候选项集供考虑。另一方面，Apriori只生成并计算一个项目集，即f1 3 4 5g，因为它先验地断定其他组合不可能有最小支持度。

*正确性 我们需要证明，Ck Lk 。*显然，一个大项目集的任何子集也必须有最小支持度。因此，如果我们用所有可能的项目扩展Lk\_l中的每个项目集，然后删除所有那些(k-1)-子集不在Lk\_l中的项目集，我们就会剩下Lk中项目集的超集。

*这个连接相当于用*数据库中的*每个*项目*扩展Lk\_l*，然后删除那些通过删除第(k-1)个项目而得到的(k-1)个项目集不在Lk\_l中的项目集。 条件p.itemk\_l

*< q.itemk\_l只是确保没有重复的*

*生成。 因此，在连接步骤之后，Ck Lk.根据*类似的推理，在修剪步骤中，我们从Ck中删除其(k-1)子集不在Lk\_l中的所有项目集，也不会删除任何可能在Lk中的项目集。

*变化。 一次计算多个大小的候选者 在第k次计算中，我们可以不只计算大小为k的候选者，还可以*

*k l*

*低于扫描数据库的费用。*

*2.1.2子集功能*

*候选项目集Ck被存储在一个哈希树中。*散列树的*一个*节点要么包含一个项目集的列表（叶子节点），要么包含一个散列表（内部节点）。在一个内部节点中，哈希表的每个桶都指向另一个节点。哈希树的根被定义为在深度1。深度为d的内部节点指向深度为d+1的节点。项目集被存储在叶子里。当我们添加一个项目集c时，我们从根部开始，沿着树往下走，直到我们到达一个叶子。在深度为d的内部节点，我们通过对项目集的第d个项目应用哈希函数来决定遵循哪个分支。所有节点最初都被创建为叶子节点。当叶子节点中的项目集数量超过一个指定的阈值时，叶子节点就会被转换为内部节点。

*从根节点开始，子集函数*

*找到包含在交易*t*中的所有候选人，*方法如下。如果我们在一个叶子上，我们找到叶子中的哪些项目集包含在t中，并将它们的引用添加到答案集中。如果我们在一个内部节点，并且我们通过散列项目i到达该节点，我们对t中i之后的每个项目进行散列，并递归地将此程序应用于相应桶中的节点。对于根节点，我们对t中的每个项目进行散列。

*为了了解为什么子集函数会返回所需的*引用集，请考虑在根节点发生了什么。对于包含在交易t中的任何项目集c，c的第一个项目必须在t中。在根节点，通过对t中的每个项目进行散列，我们确保我们只忽略以不在t中的项目开始的项目集。唯一的额外因素是，由于任何项目集中的项目都是有序的，如果我们通过散列项目i到达当前节点，我们只需要考虑t中发生在i之后的项目。

*2.2算法 AprioriTid*

*AprioriTid算法，如图2所示，也是*使用apriori-gen函数（在第2.1.1节中给出），在传递开始前确定候选项集。这个算法有趣的地方在于，数据库D在第一次传递后不用于计算支持度。 相反，集合Ck被用于这一目的。集合Ck的每个成员都是< T ID, fXk g >的形式，其中每个Xk是存在于具有标识符TID的交易中的一个潜在的大k-项目集。 对于k = 1，Cl对应于

*计算候选人C/*

*k l*

*，其中C/*

*产生*

*数据库D，尽管在概念上每个项目 i*

*从C ，等等。 请注意，C/ C*

*因为Cis*



*是由项目集fig代替的。 对于k>1，C是*

*kk l*

*k l*

*k lk*

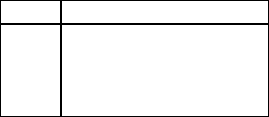
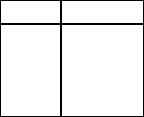
*从Lk生成的。这种变化可以在算法（步骤10）产生的 结果中得到回报。该成员*



*的Ck对应于交易t是<t.T ID, fc* E Ckjc包含在tg>。 如果一个交易不包含任何候选的k-项目集，那么Ck就不会有这个交易的条目。因此，Ck中的条目数可能小于数据库中的交易数，特别是对于大的k值而言。此外，对于大的k值而言，每个条目可能小于相应的交易，因为交易中可能包含极少的候选项目。然而，对于小的k值，每个条目可能大于相应的交易，因为Ck中的条目包括交易中包含的所有候选k-项目集。

*在第2.2.1节中，我们给出了用于*实现该算法的*数据结构*。关于正确性的证明和缓冲区管理的讨论见5]。

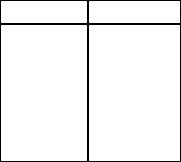
*我们用L3生成C4，结果发现它是空的，*我们就终止了。

*数据库 C1*

|  |  |  |  |
| --- | --- | --- | --- |
| *TID* | *项目* | *TID* | *项目集的集合* |
| *100* | *1 3 4* | *100* | *f f1g, f3g, f4g g* |
| *200* | *2 3 5* | *200* | *F F2G, F3G, F5G G* |
| *300* | *1 2 3 5* | *300* | *f f1g, f2g, f3g, f5g g* |
| *400* | *2 5* | *400* | *f f2g, f5g g* |

*C2*

*L1*



*项目集*

*f1 2g*

*f1 3g*

*f1 5g*

*f2 3g*

*f2 5g*

*脂肪含量为3.5g*

*支持*

*1*

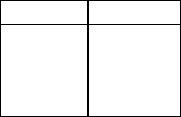
*2*

*1*

*2*

*3*

*2*



*项目集*

*F1G F2G F3G F5G*

*支持*

*2*

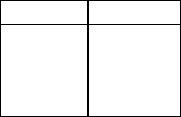
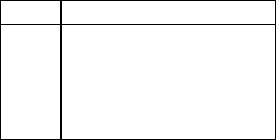
*3*

*3*

*3*



*1) L1 = flarge 1-itemsetsg;*



*C2*

*TID*

*100*

*200*

*300*

*400*

*项目集的集合*

*f f1 3g g*

*F F2 3G, F2 5G, F3 5G G F1 2G, F1 3G, F1 5G, F2 3G, F2 5G, F3 5G G*

*f f2 5g g*

*L2*

*项目集*

*f1 3g*

*f2 3g*

*f2 5g*

*脂肪含量为3.5g*

*支持*

*2*

*2*

*3*

*2*

*2）C1=数据库D。*

*3) for ( k = 2; Lk-1 0; k++ ) 做开始*

*4) Ck = apriori-gen(Lk-1 ); 新候选人*

*5) Ck = 0。*



*6) 对于所有条目t E C*

*k-1*

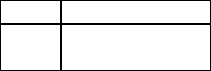
*做开始*

 *7)确定Ck中包含的 候选项集*

*Ct = fc E Ck*

*C3*

*j (c - c[k]) E. set-of-itemsets / t. C3*



*TID*

*200*

*300*

*项目集的集合*

*f f2 3 5g g*

*f f2 3 5g g*

*在交易中的标识符t.TID*

*(c-c[k-1]) E t.set-of-itemsetsg。*

*8) forall candidates c E Ct do*

*9) c. count++。*



*10) 如果（Ct = 0），那么Ck += < t. TID, Ct >。*

*11）结束*

*12)Lk = fc E Ck j c. count 2: minsupg*

*13) 结束*

*14)答案=Sk Lk ;*

*图2：算法AprioriTid*

*例子 考虑图3中的数据库，*假设最小支持度为2个交易。在步骤4用Ll调用apriori-gen得到候选项集C2。在步骤6到10中，我们通过迭代Cl中的条目来计算C2中候选项的支持度，并生成C2。 Cl中的第一个条目是f f1g f3g f4g g，对应于交易

*100. 在步骤7的Ct对应于这个条目t*

*是f f1 3g g，因为f1 3g是C2的成员，而且*（f1 3g - f1g）和（f1 3g - f3g）都是t.set-of-itemsets的成员。

*用L2调用apriori-gen得到C3。*用C2和C3对数据进行*传递*，产生C3。请注意



*项目集支持*

*f2 3 5g*

*2*

*L3*



*项目集支持*

*f2 3 5g*

*2*

*图3：示例*

*2.2.1数据结构*

*我们给每个候选项目集分配一个唯一的数字，*称为它的ID。每个候选项集Ck被保存在一个数组中，以Ck中的项集的ID为索引。Ck的一个成员现在的形式是< TID, fIDg >。每个Ck被存储在一个顺序结构中。

*apriori-gen函数*通过连接两个大的(k-1)项目集来*生成一个候选的k*-项目集ck。 我们为每个候选项目集维护两个额外的字段：i）生成器和ii）扩展。候选项集ck的生成器字段存储了两个大（k-1）项集的ID，这两个项集的连接产生了ck。 项目集ck的扩展字段存储了所有作为ck扩展的(k + 1)-候选项的ID。 因此，当一个候选项ck是

*k\_l*

*k\_l*

*在C3中没有与TID有关的交易条目*

*通过连接ll产生*

*和l2*

*，我们保存ID*

*100和400，因为它们不包含任何的*

*k\_l*

*l*

*k\_l*

*l的*

*和l2*

*在生成器领域为ck。 在*

*C3中的项目集。C3中的候选项f2 3 5g变成了*

*出大，是L3的唯一成员。当*

*同时，ck的ID被添加到扩展名中。*

*领域 .*

*k\_l*

*现在我们描述一下图2的步骤7是如何*使用上述数据结构实现的。回顾一下，Ck\_l中的条目t.set-of-itemsets字段给出了交易t.TID中包含的所有（k-1）候选项的ID。对于每一个这样的候选项ck\_l，扩展字段给出了Tk，即所有作为ck\_l扩展的候选项k-itemsets的ID集合。 对于Tk中的每个ck，生成器字段给出生成ck的两个项目集的ID。 如果这些项目集存在于t.set-of-itemsets的条目中，我们可以得出结论，ck存在于交易t.TID中，并将ck加入Ct。

# *3个性能*

*为了评估这些算法*在发现大型数据集方面的*相对性能*，我们在一台IBM RS/6000 530H工作站上进行了几次实验，该工作站的CPU时钟频率为33MHz，主内存为64MB，运行AIX 3.2。数据驻留在AIX文件系统中，存储在一个2GB的SCSI 3.5 "驱动器上，测量的连续吞吐量约为2MB/秒。

*我们首先对AIS 4]和SETM进行了概述。*

*13]算法，与之相比，我们比较了*Apriori和AprioriTid算法的性能。 然后，我们描述了在性能评估中使用的合成数据集，并展示了性能的重新结果。最后，我们描述了如何将Apriori和AprioriTid的最佳性能特征结合到AprioriHybrid算法中，并展示了其扩展特性。

*3.1 AIS算法*

在扫描数据库时，*候选项目集被生成并被*快速计算。在读取一个交易后，确定哪些在前一次扫描中被发现是大的项目集包含在这个交易中。新的候选项目集是通过用交易中的其他项目扩展这些大项目集而产生的。一个大的项目集l只用那些大的项目来扩展，并且在项目的词法排序中比l中的任何项目都要晚出现。关于AIS算法的进一步细节，见4]。

*3.2 SETM算法*

*SETM算法13]的动机是希望*使用SQL来计算大项目集。 和AIS一样，SETM算法也是基于从数据库中读取的事务，即时生成候选人。

*因此，它生成并统计*AIS算法生成的*每个候选项集*。然而，为了使用标准的SQL连接操作来生成候选项，SETM将候选项的生成与计数分开。它将候选项目集的副本与生成事务的TID一起保存在一个顺序结构中。在传递结束时，候选项目集的支持计数是通过对这个顺序结构进行排序和汇总来确定的。

*SETM记住了*具有候选项目集的*生成*事务的*TID*。为了避免需要进行子集操作，它使用这些信息来确定读到的交易中包含的大项集。Lk C Ck，并通过删除那些不具有最小支持度的候选项而得到。假设数据库是按TID顺序排序的，SETM可以在下一次通过TID对Lk进行排序，轻松找到交易中包含的大项集。事实上，它只需要在TID顺序中访问Lk的每个成员一次，而候选者的生成可以使用关系型合并-连接操作来进行

*13].*

*这种方法的缺点主要是*由于候选集Ck的大小问题。对于每个候选项集，候选项集现在有与候选项集出现的交易数量一样多的条目。此外，当我们准备在通证结束时计算候选项目集的支持度时，Ck的顺序是错误的，需要对项目集进行排序。在计算和修剪掉不具有最小支持度的小的候选项集后，得到的集合Lk需要在TID上再进行一次排序，然后才能用于在下一环节生成候选项。

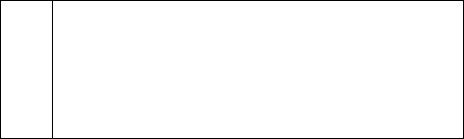
*3.3 合成数据 的生成*

*我们生成了合成交易，以评估算法*在大量数据特征下的性能。这些交易模仿了零售环境中的交易。我们对 "真实 "世界的模型是，人们倾向于一起购买成套的物品。每个这样的集合都可能是一个最大的大物品集。这样一个套装的例子可能是床单、枕套、被子和围脖。然而，有些人可能只购买这样一个套装中的一些物品。例如，有些人可能只买床单和枕套，而有些人只买床单。一个交易可能包含一个以上的大物品集。例如，一个顾客在订购床单和枕头套时，可能会订购一件衣服和夹克，而衣服和夹克一起构成了另一个大物品集。交易规模通常集中在一个平均值周围，少数交易有许多项目。大项目集的典型尺寸也是

*聚集在一个平均值周围，少数大项目*集有大量的项目。

*为了创建一个数据集，我们的合成数据生成*程序采取了表 *2*中的参数。

*表2：参数*



*jDj*

*jT jIj*

*jLj N*

*交易数量*

*交易的平均规模 最大的潜在大项目集的平均规模*

*最大的潜在大项目集的数量 项目的数量*

*我们首先确定下一笔交易的规模。*注意，如果每个项目以相同的概率p选择，并且有N个项目，那么交易中的预期项目数由参数为N和p的二项分布给出，并由平均数为N p的泊松分布近似。

*然后，我们将项目分配给交易。每个*交易都被分配了一系列潜在的大项目集。如果手头的大项目集不适合放在交易中，那么在一半的情况下，项目集还是被放在交易中，而在其余情况下，项目集被移到下一个交易中。

*大项目集是从这样的*项目集*集合T中选择的。*T中项目集的数量被设定为jLj。jLj和潜在的大项目集的平均支持度之间有一个反比关系。T中的项目集是通过首先从均值μ等于jIj的泊松分布中挑选项目集的大小而产生的。 第一个项目集中的项目是随机选择的。为了模拟大项目集经常有共同的项目这一现象，在随后的项目集中，有一部分项目是从之前生成的项目集中选择的。我们使用一个指数分布的随机变量，其平均值等于相关水平，以决定每个项目集的这部分内容。剩下的项目是随机挑选的。在实验中使用的数据集中，相关水平被设置为0.5。我们进行了一些实验，将相关水平设置为0.25和

*0.75，但没有发现*我们的表现结果的*性质有太大的不同*。

*T中的每个项目集都有一个与之相关的权重*，它对应于这个项目集被选中的概率。这个权重是从具有单位平均数的指数分布中挑选出来的，然后被归一化，使T中所有项目集的权重之和为1。 下一个要放入交易的项目集是通过抛掷一枚jLj面的加权硬币从T中选择的，其中一面的权重是

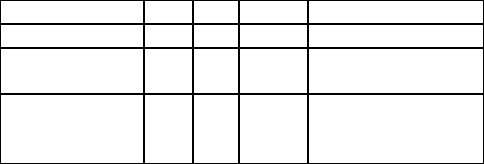
*挑选相关项目集的概率。*

*为了模拟*一个大的物品集中*的所有物品*并不总是一起买的*现象*，我们给T中的每个物品集分配一个腐败级别c。当把一个物品集添加到交易中时，只要0到1之间的均匀分布的随机数小于c，我们就一直从物品集中删除一个物品。一个项目集的腐败程度是固定的，并从均值 为*0.5的*正态分布中得到。

*和 方差0.1。*

*我们通过设置N=1000和jLj来生成数据集。*

*= 2000.我们为jT j选择了3个值：5、10和20。我们*还为jIj选择了3个值：2、4和6。交易数被设定为100,000，因为正如我们将在第3.4节中看到的，SETM无法运行更大的数值。然而，在我们的扩展实验中，我们产生了高达1000万个交易的数据集（T20为838MB）。表3总结了数据集的参数设置。对于相同的jT j和jDj值，不同的jIj值，数据集的大小以兆字节为单位，大致相等。

*表3：参数设置*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *命名* | *jT j* | *jIj* | *jDj* | *尺寸以兆字节为单位* |
| *T5.I2.D100K* | *5* | *2* | *100K* | *2.4* |
| *T10.I2.D100K* | *10* | *2* | *100K* | *4.4* |
| *T10.I4.D100K* | *10* | *4* | *100K* |  |
| *T20.I2.D100K* | *20* | *2* | *100K* | *8.4* |
| *T20.I4.D100K* | *20* | *4* | *100K* |  |
| *T20.I6.D100K* | *20* | *6* | *100K* |  |

*3.4 相对性能*

*图4显示了*表3中给出的*六个*合成数据集的*执行时间*，最小支持度的数值越来越小。随着最小支持度的降低，所有算法的执行时间都在增加，因为候选项集和大项集的总数在增加。

*对于SETM，我们只绘制了执行的*

*图4为T5.I2.D100K数据集的时间。*表 *4*给出了SETM在两个平均交易规模为10的数据集上*的*执行时间。

*我们没有在表 4中绘制执行时间。*

*因为*与其他算法的执行时间相比，*它们太大了*。对于交易规模为20的三个数据集，SETM的执行时间过长，由于趋势明显，我们放弃了这些运行。显然，对于大数据集来说，Apriori胜过SETM不止一个数量级。

*t5.i2.d100kt10.i2.d100k*

80160

医学博士 AIS

AprioriTid Apriori

AIS

AprioriTid Apriori

70140

60120

50100

时间（秒

时间（秒

4080

3060

2040

1020

0

21.5

10.75

0.5

0.33

0.25

0

21.5

10.75

0.5

0.33

0.25

最低限度的支持 最低限度的支持

350

300

250

200

时间（秒

150

100

50

0

*t10.i4.d100kt20.i2.d100k*

1000

AIS

AprioriTid Apriori

AIS

AprioriTid Apriori

900

800

700

600

时间（秒

500

400

300

200

100

0

21.5

10.75

0.5

0.33

0.25

21.5

10.75

0.5

0.33

0.25

最低限度的支持 最低限度的支持

1800

1600

1400

1200

时间（秒

1000

800

600

400

200

0

*t20.i4.d100kt20.i6.d100k*

3500

AIS

AprioriTid Apriori

AIS

AprioriTid Apriori

3000

2500

2000

时间（秒

1500

1000

500

0

21.5

10.75

0.5

0.33

0.25

21.5

10.75

0.5

0.33

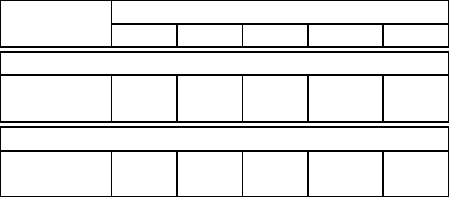
0.25

最低限度的支持

*图4：执行时间*

最低限度的支持

*表4：SETM的执行时间（秒）。*



*算法*

*最低限度的支持*

*2.0%1.5%1.0%0.75%*

*数据集T10.I2.D100K*

*0.5%*

*数据集T10.I4.D100K*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *ǞǞSETM* | *74* | *161* | *838* | *1262* | *1878* |
| *Apriori* | *4.4* | *5.3* | *11.0* | *14.5* | *15.3* |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *ǞǞSETM* | *41* | *91* | *659* | *929* | *1639* |
| *Apriori* | *3.8* | *4.8* | *11.2* | *17.4* | *19.3* |

*Apriori在所有的问题规模上都战胜了AIS，系数*从高最小支持度的2到低支持度的一个数量级以上不等。AIS总是比SETM好得多。对于小问题，AprioriTid的表现和Apriori差不多，但是对于大问题，性能下降到大约两倍的速度。

*3.5相对性能的解释*

*为了解释这些性能趋势，我们在*图5*中显示了*T10.I4.D100K数据集在最小支持度为0.75%的情况下，大集和候选集的大小。请注意，该图中的Y轴有一个对数刻度。

1e+07

C-k-m (SETM)

C-k-m (AprioriTid) C-k (AIS, SETM)

C-k (Apriori, AprioriTid)

L-k

1e+06

100000

项目集的数量

10000

1000

100

10

1

1234567

通行证号码

*图5：大集和候选集的大小*

*(T10.I4.D100K, minsup = 0.75%)*

*SETM算法的根本问题在于*

*SETM算法的表现很差。2 这解释了*表4中SETM在交易规模为10的数据集上从1.5%的支持率到1.0%的支持率时的时间跳跃。13]中SETM的扩展实验中最大的数据集仍然足够小，Ck可以装入内存；因此他们没有遇到这种执行时间的跳跃。请注意，在相同的最小支持度下，候选项集的支持度随着交易数量的增加而线性增加。因此，当我们在jT j和jIj值相同的情况下增加交易数量时，虽然Ck的大小没有变化，但Ck的大小会线性上升。因此，对于有更多交易的数据集，SETM和其他算法之间的性能差距将变得更大。

*AIS的问题是，它产生了太多的*

*候选者，后来发现这些候选者很小，导致*它浪费了太多的精力。Apriori在第二遍中也计算了太多的小集合（记得C2实际上是Ll与Ll的交叉产物）。 然而，从第三遍开始，这种浪费就急剧减少了。请注意，对于图5中的例子，在第三遍之后，几乎每一个被Apriori计算的候选项集都变成了一个大集。

*AprioriTid也有SETM的问题，即Ck倾向于*很大。然而，AprioriTid所使用的priori候选人生成方式比SETM所使用的基于交易的candi-date生成方式所产生的候选人要少得多。因此，AprioriTid的Ck比SETM的要少。Apri- oriTid也能够使用一个词（ID）来存储一个候选者，而不是需要与候选者中的项目数量一样多的词。3 此外，与SETM不同，AprioriTid不需要对Ck进行排序。因此，AprioriTid并不像SETM那样因维护Ck而受到影响。

*AprioriTid有一个很好的特点，它*通过对Ck集的传递来取代对原始数据集*的*传递。 因此，当Ck的大小变得比数据集小时，AprioriTid在后面的传递中非常有效。



*2 在SETM中，外部排序的成本可以得到一定程度的降低，具体如下。在把Ck中的条目写到磁盘之前，我们可以用内部排序的方式对它们进行项目集排序*



*计算方法是其Ck的大小，*集合Ck的大小由以下公式给出

*X*

*集。回顾一下*

*支持数(c)。*

*程序，并把它们写成排序的运行。这些排序后的运行*

*然后可以合并以获得支持度计数。然而，鉴于SETM的性能较差，我们并不期望这种优化会影响算法的选择。*

*3 为了让SETM使用ID，它必须在内存中维护两个额外的数据结构 一个哈希表来寻找*

*候选项目集c*



*因此，Ck集大约比*相应的Ck集*大S倍*，其中S是候选项集的平均支持数。除非问题规模非常小，否则Ck集必须被写入磁盘，并在外部进行两次排序，导致

*出了一个候选人是否以前产生过，以及*

*一个从ID到候选人的映射。然而，这将破坏该算法的面向集合的性质。另外，一旦我们有了给我们提供候选者ID的哈希表，我们不妨同时计算它们，以避免两种外部排序。我们对SETM的这个变体进行了实验，发现虽然它比SETM做得更好，但它的表现仍然比Apriori或AprioriTid差很多。*

*数据库的大小。因此，我们发现，*当*AprioriTid*的Ck集可以容纳在内存中，并且大项集的分布有一个长尾时，*AprioriTid*击败了Apriori。当Ck不适合放在内存中时，AprioriTid的执行时间就会出现跳跃，例如当交易规模从0.75%到0.5%的数据集时

*图4中的10。 在这个区域，Apriori开始跳动*

*AprioriTid.*

*3.6算法AprioriHybrid*

*没有必要在所有的*数据传递中*使用相同的算法*。图6显示了Apriori和AprioriTid在数据集T10.I4.D100K上的不同传递的执行时间。在早期的处理中，Apriori比AprioriTid做得更好。然而，AprioriTid在后来的过程中击败了Apriori。我们观察到其他数据集有类似的相对行为，其原因如下。 Apriori和AprioriTid使用相同的候选人生成程序，因此计算相同的项目集。在后面的程序中，候选项集的数量减少了（见图5中Apriori和AprioriTid的Ck大小）。然而，Apriori仍然检查数据库中的每一个交易。另一方面，AprioriTid不是扫描数据库，而是扫描Ck以获得支持数，Ck的大小已经变得比数据库的大小小。当Ck集可以容纳在内存中时，我们甚至不会产生将它们写入磁盘的成本。

14

Apriori AprioriTid

12

10

8

时间（秒

6

4

2

0

1234567

通过 #

*图6：Apriori和Burberry的单次执行时间。*

*AprioriTid (T10.I4.D100K, minsup = 0.75%)*

*基于这些观察，我们可以设计一种*混合算法，我们称之为AprioriHybrid，它在最初的传递中使用Apriori，并在预期传递结束时的集合Ck将适合内存时切换到AprioriTid。我们使用以下启发式方法来估计Ck在下一次传递中是否适合内存。在当前传递结束时，我们有候选者的计数



*在Ck中。 由此，我们估计*如果*Ck*被生成，它*的规模*会是*多少。* 这

*规模，换句话说，就是（Pcandidates c E Ck support(c) +*

*交易的数量）。)如果这一次的Ck小到*足以容纳在内存中，并且在当前通道中的大的候选者比上一次的少，我们就切换到AprioriTid。添加后一个条件是为了避免在当前程序中的Ck适合内存，但下一个程序中的Ck可能不适合时进行切换。

*从Apriori切换到AprioriTid确实涉及到*一个成本。假设我们决定在第k遍结束时从Apriori切换到AprioriTid。 在第(k+1)次传递中，在找到交易中包含的候选项目集后，我们还必须将它们的ID添加到Ck l中（见2.2节中对AprioriTid的描述）。因此，相对于仅仅运行Apriori而言，在这个环节中会产生额外的成本。只有在第(k+2)个环节，我们才真正开始运行AprioriTid。因此，如果没有大的(k+1)项目集，或者没有(k+2)候选项目，我们就会产生转换的成本，而没有得到使用AprioriTid的任何节省。

*图7显示了AprioriHybrid的性能。*

*相对于Apriori和AprioriTid的三个数据集，AprioriHybrid在几乎所有情况下都比Apriori表现好。*AprioriHybrid几乎在所有情况下都比Apriori表现得好。对于支持率为1.5%的T10.I2.D100K，AprioriHybrid的表现比Apriori稍差，因为发生切换的那一关是最后一关；AprioriHybrid因此产生了切换的成本，但没有实现其好处。一般来说，AprioriHybrid相对于Apriori的优势取决于Ck集的大小在后面的过程中如何下降。如果Ck一直大到接近尾声，然后有一个突然的下降，我们使用AprioriHybrid就不会有什么收获，因为我们只能在切换后的短时间内使用AprioriTid。这就是T20.I6.D100K数据集的情况。 另一方面，如果Ck的大小逐渐下降，AprioriTid可以在切换后使用一段时间，在执行时间上可以得到明显的改善。

*3.7 规模化实验*

*图8显示了AprioriHybrid是如何在*交易数量从100,000个增加到100,000个时扩大规模的。

*1000万笔交易。 我们使用的组合是*

*(T5.I2), (T10.I4), 和 (T20.I6)*分别*代表*交易和项目集的*平均规模*。所有其他参数都与表 *3*中的数据相同*。*

*1000万笔交易的这些数据集的大小*分别为239MB、439MB和838MB。最小支持度被设定为0.75%。执行时间相对于第一个10万个交易数据集的时间来说是正常化的。

*T10.I2.D100K*

40

AprioriTid Apriori AprioriHybrid

35

30

*图，以及*第二张图中的*100万交易*数据集。如图所示，执行时间的比例是相当线性的。

12

T20.I6 T10.I4 T5.I2

10

258

时间（秒

相对时间

20

6

15

4

10

52

0

21.5

10.75

0.5

0.33

0. 250

最低限度的支持

1002505007501000

交易数量（以'000计

*T10.I4.D100K*

55

AprioriTid Apriori

AprioriHybrid

50

45

40

35

时间（秒

30

25

20

15

10

5

0

2

1.5

10.75

0.5

0.33

0.25

14

12

T20.I6 T10.I4 T5.I2

10

相对时间

8

6

4

2

0

12.557.510

交易数量（以百万计

*T20.I6.D100K*

700

AprioriTid Apriori AprioriHybrid

600

500

400

时间（秒

300

200

100

最低限度的支持

*图8：交易数量的扩展* 接下来，我们研究了AprioriHybrid是如何扩展的

*随着项目数量的增加。 我们增加了*

*在*T5.I2.D100K、T10.I4.D100K和T20.I6.D100K*三个*参数设置下，*项目数从1000到10,000*。所有其他参数都与表3中的数据相同。我们对最小支持度为0.75%进行了实验，得到了图9所示的结果。由于项目的平均支持度随着项目数量的增加而减少，所以执行时间稍有减少。 这导致了大项目集的减少，因此，执行速度更快。

0

21.5

10.75

0.5

0.33

0.25

*次。*

最低限度的支持

*图7：执行时间。AprioriHybrid*

*最后，我们调查了扩大规模的问题，因为我们增加了*

*的平均交易规模。这个*实验的*目的*是看我们的数据结构是如何随着交易规模的变化而变化的，与其他因素如物理数据库规模和大项目集的数量无关。 我们保持数据库的物理大小

45

T20.I6 T10.I4 T5.I2

40

35

30

时间（秒

25

20

15

10

5

0

100025005000750010000

项目数量

30

25

500

750

1000

20

时间（秒

15

10

5

0

51020304050

交易规模

*图9：项目数量的扩大*

*通过保持*平均交易规模和交易数量的*乘积*不变，*数据库的交易数量大致不变*。交易数量从平均交易规模为5的数据库的200,000到平均交易规模为50的数据库的20,000不等。将最小支持度固定为一个百分比会导致大项目集的数量随着交易规模的增加而大量增加，因为一个项目集出现在交易中的概率与交易规模大致成正比。因此，我们以交易的数量来固定最小支持度。结果显示在图10中。关键词中的数字（如500）指的是这个最小支持度。如图所示，执行时间随着交易规模的增加而增加，但只是逐渐增加。增加的主要原因是，尽管在交易数量方面设置了最小支持度，但大项目集的数量随着交易长度的增加而增加。次要的原因是，寻找交易中存在的候选者需要更长的时间。

# *4结论和未来工作*

*我们提出了两种新的算法，Apriori和Apri-* oriTid，用于发现大型交易数据库中项目之间的所有重要关联规则。我们将这些算法与之前已知的算法，即AIS 4]和SETM 13]算法进行了比较。我们提出了实验结果，表明所提出的算法总是优于AIS和SETM。性能差距随着问题的大小而增加，小问题的差距是3倍，大问题的差距超过一个数量级。

*我们展示了这两个专业的最佳特征是怎样的。*

*图10：交易规模的扩大*

*这些算法可以组合成一个混合算法*，称为AprioriHybrid，然后成为这个问题的首选算法。扩展实验表明，AprioriHybrid随着交易数量的增加而线性扩展。此外，随着数据库中项目数量的增加，执行时间也略有下降。随着平均交易规模的增加（同时保持数据库规模不变），执行时间只逐渐增加。这些实验证明了在涉及非常大的数据库的实际应用中使用AprioriHybrid的可行性。

*本文介绍的算法已经*在几个数据存储库上实现，包括AIX文件系统、DB2/MVS和DB2/6000。 我们还用真实的客户数据对这些算法进行了测试，其细节可以在以下文章中找到

*5].在未来，我们计划从*以下几个方面来*扩展这项工作*。

* *项目上的多个分类法（is-a层次结构）往往是可用的。这种层次结构的一个例子是，洗碗机是厨房电器是重型电器，等等。我们希望能够找到使用这种层次结构的关联规则。*
* *我们没有考虑交易中购买的物品的数量，这对某些应用是有用的。寻找这样的规则需要进一步的工作。*

*本文报告的工作是*在IBM Al-maden研究中心的Quest项目背景下*完成的*。在Quest中，我们正在探索数据库挖掘问题的各个方面。除了发现关联规则的问题，我们研究的其他一些问题包括

*通过分类*查询2]和时间序列的相似性查询1]来*增强数据库能力。*我们认为，数据库挖掘是数据库的一个重要的新的应用领域，它将商业利益和耐人寻味的研究问题结合起来。

*鸣谢 我们要感谢Mike Carey提出的有见地的*意见和建议。

# *参考文献*

*1]R. Agrawal, C. Faloutsos, and A. Swami.Ef-ficient similarity search in sequence databases.In Proc. of the Fourth International Conference on Foundations of Data Organization and Algo- rithms, Chicago, October1993.*

*2]R. Agrawal, S. Ghosh, T. Imielinski, B. Iyer, and*

*A.Swami.一个用于数据库*挖掘应用的区*间分类器*.In Proc. of the VLDB Conference, pages 560-573, Vancouver, British Columbia, Canada, *1992.*

*3]R. Agrawal, T. Imielinski, and A. Swami.数据库挖掘。A performance perspective.IEEE Transactions on Knowledge and Data En- gineering, 5(6):914-925, December 1993.基于知识的数据库中的学习和发现特刊。*

*4]R. Agrawal, T. Imielinski, and A. Swami.挖掘大型数据库中项目集之间的关联规则。In Proc. of the ACM SIGMOD Con- ference on Management of Data, Washington, D.C., May1993.*

*5]R. Agrawal and R. Srikant.*在大型数据库中挖掘关联规则*的快速算法*。Re-search Report RJ 9839, IBM Almaden Research Center, San Jose, California, *June1994.*

*6] D. S. Associates.The new direct marketing.*

*Business One Irwin, Illinois,1990.*

*7]R. Brachman等人，数据考古学的综合支持。In AAAI-93 Workshop on Knowledge Discovery in Databases, July1993.*

*8]L. Breiman, J. H. Friedman, R. A. Olshen, and*

*C.J. Stone.Classification and Regression Trees.Wadsworth, Belmont, 1984.*

*9]P. Cheeseman等人，Autoclass。A bayesian classification system.In 5th Int l Conf. on Machine Learning.Morgan Kaufman, June1988.*

*10]D. H. Fisher.通过增量概念聚类的知识获取*。机器学习, 2(2), *1987.*

*11]J. Han, Y. Cai, and N. Cercone.*数据库中的*知识*发现。一个面向属性的方法。In Proc. of the VLDB Conference, pages 547-559, Vancouver, British Columbia, Canada, *1992.*

*12]M. Holsheimer and A. Siebes.数据挖掘。*在数据库中寻找知识。技术报告CS-R9406，CWI，荷兰， *1994。*

*13]M. Houtsma and A. Swami.面向集合的*关联规则的*挖掘*。研究报告RJ 9567，IBM Almaden研究中心，加州圣何塞，*1993年*10月 *。*

*14]R. Krishnamurthy and T. Imielinski.Practi-* tioner problems in need of database research:知识发现中的再搜索方向。SIG- MOD RECORD, 20(3):76-78, *September1991.*

*15] P. Langley, H. Simon, G. Bradshaw, and*

*J.Zytkow.科学发现:创造性过程的计算探索》。麻省理工学院出版社，1987年。*

*16]H. Mannila and K.-J. Raiha.依赖性推理.In Proc. of the VLDB Conference, pages 155-158, Brighton, England, 1987.*

*17]H. Mannila, H. Toivonen, and A. I. Verkamo. 发现关联规则的有效算法。In KDD-94: AAAI数据库中的知识发现研讨会，1994年7月 。*

*18]S. Muggleton和C. Feng.*逻辑程序的*有效归纳*。In S. Muggleton, editor, Inductive Logic Programming. Academic Press, 1992.

*19]J. Pearl.智能*系统*中的概率推理*: 合理推理的 网络, *1992.*

*20] G. Piatestsky-Shapiro.* 强势规则的*发现、分析*和展示.在

*G. Piatestsky-Shapiro，编辑，数据库中的知识发现。AAAI/MIT Press, 1991.*

*21]G. Piatestsky-Shapiro, editor.数据库中的知识发现.AAAI/MIT Press, 1991.*

*22]J. R. Quinlan.C4.5: 机器学习的程序.Morgan Kaufman, 1993.*