一种基于矩阵的关联规则挖掘新算法*)

丁艳辉　王洪国　高　明　谷建军
(山东师范大学信息管理学院　济南 250014)

摘　要　本文针对大型交易事务数据库数据间发现关联规则问题，提出了一个新的关联规则挖掘算法，BOM(Base On Matrix)算法。该算法不同于经典的 Apriori 算法，对于大型交易事务数据库，具有较 Apriori 算法更加优越的性能。

关键词　大型交易事务数据库，矩阵，关联规则

A New Algorithm Based on Matrix for Mining Association Rules

DING Yan-Hui　WANG Hong-Guo　GAO Ming　GU Jian-Jun
(Information Management School of Shandong Normal University , Jinan 250014)

Abstract　In this paper, the problem of discovering association rules between items in a large database of sales transactions is discussed, and a novel algorithm , BOM (Base On Matrix), is proposed. The Proposed algorithm is fundamentally different from the known algorithm Apriori. Empirical evaluation shows that the algorithm outperforms the known one for large databases.

Keywords　Large database of sales transactions, Matrix, Association rules

1　引言

随着数据库技术的进步，零售组织可以方便地收集和存储大量的销售数据。这些数据记录中包含了交易的时间和在这次交易中购买的商品[1,2]，包含着一些潜规则有待于进一步挖掘并在实践中应用。经典的关联规则挖掘算法 Apriori 算法[3,4]对于大型交易事务数据库的应用有一定的局限性，因此，有必要研究提出一种新的实用有效算法取代之。

关联规则挖掘问题的形式化描述[2]如下：

设 $I=(i_1,i_2,\cdots,i_m)$为数据项的集合，任务相关的数据 D 是数据库事务的集合，其中每个事务 T 是数据项的集合，使得 $T\subseteq I$，关联规则是形如 $A\Rightarrow B$ 的蕴涵式，其中 $A\subset I$，$B\subset I$，并且 $A\subset B=\phi$。定义支持度(support)为 D 中包含 $A\cup B$ 的百分比，置信度(confidence)为 D 中包含 A 的事务同时也包含 B 的百分比。

项的集合称为项集，包含 k 个项的项集称为 k-项集。如果项集满足最小支持度 minsup，即项集的出现频率大于或等于 minsup 与 D 中事务总数的乘积，则称它为频繁项集(Frequent itemset)。

0-1 矩阵是一种很好的数据表示形式。BitMatrix 算法[5]中也采用了利用矩阵来发现关联规则，但是 BitMatrix 算法仅把矩阵作为数据库数据的存储形式，没有利用矩阵的直接运算特性，算法思想仍然基于 Apriori 算法，执行过程中仍然要产生大量的候选项集，且需要依次获得各级频繁项集。本文提出了一个新的关联规则挖掘算法，BOM(Base On Matrix)算法，不但使用矩阵来表示基础数据，而且通过矩阵的直接运算直接得到频繁 k-项集，避免了生成大量的候选项集，提高了对大型交易数据库的处理效率。

文章第 2 部分给出 BOM 算法，用于发现频繁项集；第 3 部分对 BOM 算法与 Apriori 算法进行比较。

2　BOM 算法

BOM 算法的思想是：首先，扫描数据库，生成对应的初始矩阵；其次，利用分析得到的频繁项集的大小 k，对矩阵进行修剪；然后，直接生成频繁 k-项集。

2.1　算法设计

Apriori 算法中的每一事务中的数据项要以字典顺序排序[4]。而 BOM 算法中，不需要进行排序，通过编程就可以使在频繁项集中的项目和具有相同长度的频繁项集以字典顺序排序。定义表示所有频繁 k-项数据项集。每一个 k 项数据项，包含数据 $c[1],c[2]\cdots c[k]$，$(c[1]<c[2]<\cdots<c[k])$

BOM 算法描述如下：

```
Step1: Initializing the Matrix ; //初始化矩阵
Step2: k=PreProcess_Matrix(); //对初始矩阵进行预处理
Step3: Lk=GetLargeItemsets(k); //直接获得频繁 k-项集
Step4: Answer=Lk; //返回频繁 k-项集
```

算法的第一步，按以下方式初始化矩阵：首先，构造一个矩阵，它的行和列分别代表数据项和事务。矩阵是一个位矩阵，矩阵中的每一个位置在内存中仅占一位。然后，扫描事务数据库，如果在第 j 项事务中，存在数据项 $i_1,i_2,i_3,\cdots,i_k$，那么位 $A[i_1][j]A[i_2][j],A[i_3][j],\cdots,A[i_k][j]$被初始化为 1，否则，初始化为 0。表 1 是一个事务数据库和初始化后的对应矩阵。

算法的第二步，执行 PreProcess_Matrix()过程对初始矩阵进行预处理：对每一行，每一列计数。

*)基金项目：山东省中青年科学家科研奖励基金(NO. 02BS012)。丁艳辉　硕士研究生，研究方向：数据挖掘，知识发现。王洪国　博士后，教授，硕士生导师。高　明　硕士研究生。谷建军　硕士研究生。

表 1　事务数据库及初始化后的对应矩阵

事务数据库

Tid	Itemsets
100	ABD
200	ABC
300	ABCE
400	BCE
500	AB
600	AE
700	A

初始化矩阵

数据项	T100	T200	T300	T400	T500	T600	T700
A	1	1	1	0	1	1	1
B	1	1	1	1	1	0	0
C	0	1	1	1	0	0	0
D	1	0	0	0	0	0	0
E	0	0	1	1	0	1	0

1) 删除矩阵中出现次数小于指定阈值的单个数据项。

在 BOM 算法中，频繁项集依然满足反单调性质：即频繁项集的所有非空子集必然也是频繁项集。在表 1 中，应删除 D 数据项，因为其出现次数小于用户设定的阈值与事务总数的乘积。(假设 minsup=25%)

2) 根据用户设定的支持度阈值分析得出待发现的频繁项集的大小 k。

在表 1 中，(假设 minsup=25%)，k 的合适取值为 3，因为包含 4 个数据项的事务仅出现一次，小于用户设定的支持度阈值 minsup 与事务总数的乘积。

3) 删除事务中数据项少于等于(k−1)项的事务。

因为在 BOM 算法中，不需要借助频繁(k−1)-项集获得频繁 k-项集；另外，对于超大型交易事务数据库，删除数量少于等于(k−1)的事务，一方面可以降低矩阵的大小，另一方面可以提高算法的执行效率。在表 1 中，删除事务 T500、T600、T700，修剪后的矩阵为：

修剪后的矩阵

数据项	T100	T200	T300	T400
A	1	1	1	0
B	1	1	1	1
C	0	1	1	1
E	0	0	1	1

算法的第三步，使用 GetLargeItemsets 过程来直接获得频繁 k 项集，GetLargeItemsets(k)过程的算法描述如下：

```
1) for (i1=1;i1<=m-k+1;i1++)//m 为修剪后矩阵的行数,代
     表单个数据项个数
2)   for(i2=i1+1;i2<=m-k+2;i2++)
3)       ......
4)       for(ik=i1+k-1;ik<=m; ik++)
5)           for (t=1;t<=n;t++)//n 为修剪后矩阵的列数,代
               表事务项个数
6)             count+=A[i1][t]×A[i2][t]×…A[ik][t]
7)             if(count>=minsup×|D|)then
8)               c=Pi1∪Pi2…Pik…;//Pi1,Pi2…Pik 分别为修
                   剪后矩阵 t 列 i1,i2…ik 行所对应的单个数
                   据项
9)               Lk=Lk∪c;
10)              break ;
11)            end
12)          end
13) end
14) return Lk;
```

GetLargeItemsets(k)过程的第一步到第六步，通过矩阵向量相乘后相加得到待生成 k-项集的支持度，在计算支持度时，按照如下规则：

i_1 从第一行开始，到 $m-k+1$ 结束，i_2 从第 i_1+1 开始，到 $m-k+2$ 结束，……，i_k 从 $i+k-1$ 开始，到矩阵末尾 m 结束，实现了频繁项集中的项目和具有同样长度的频繁项集以字典顺序排序。算法第七步到第十一步，判断待生成 k-项集 c 的支持度，如果该支持度满足用户设定的支持度阈值与事务总数的乘积，那么生成 k-项集 c，并将 c 加入到频繁 k-项集中。

2.2　算法正确性

需要证明在 GetLargeItemsets(k)过程中，得到了什么样的结果。在 GetLargeItemsets(k)过程中输入修剪后的位矩阵 A，如果矩阵 A 第 i 行第 j 列的值是 1，表示第 j 个事务项中包含第 i 个数据项。如果在第 j 个事务项中同时包含 k 个数据项 $i_1, i_2, \cdots, i_k$，那么 $A[i_1,j]=A[i_2,j]=\cdots=A[i_k,j]=1$。在第六步，求出所有可能的 k-项集的支持度 count，并且 k-项集以字典顺序排序；第七步，如果 count>=minsup×|D|，生成 k-项集 c，并且 $L_k=L_k\cup c$，停止本次矩阵向量相乘操作。如果 count < minsup×|D|，那么不生成该 k-项集。另外，GetLargeItemsets(k)过程也同样满足频繁项集的反单调性，即频繁项集的所有非空子集必然也是频繁项集。因为，一旦 k 个数据项在同一事务中出现，那么其超集也必然出现在这个事务中。因此，算法得到的频繁 k-项集是正确的。

2.3　一个例子

使用表 1 所表示的事务数据库。假设 minsup=25%，修剪后的初始矩阵为 A。A 的行依次代表单个数据项 A、B、C、E，列依次代表事务项 T100、T200、T300、T400

$$A=\begin{pmatrix}1&1&1&0\\1&1&1&1\\0&1&1&1\\0&0&1&1\end{pmatrix}$$

执行 BOM 算法，最终得到的频繁项集为{ABC, BCE}。

3　BOM 算法与 Apriori 算法的比较

3.1　时间复杂性分析

BOM 算法仅需扫描一次数据库，初始化矩阵后，数据库就不再被使用。其时间复杂度为 $O(nm^k)$，m 为修剪后矩阵的行数，n 为修剪后矩阵的列数，k 为待发现的频繁项集的大小；Apriori 算法在执行过程中，需要多次扫描数据库，算法的时间复杂性为 $O(|D|^{k+1})$，$|D|$ 为交易事务数据库中包含的交易数量。在实际应用中，大型交易事务数据库中交易的数量一般远大于商品的数量，即 $|D|>>m$。所以，BOM 算法具有较 Apriori 算法好的时间特性。

3.2　空间复杂性分析

Apriori 算法会产生大量的候选项集，并且所有具有相同长度的候选项集必须被存储在内存中，这将会占用大量的内存空间，导致内存不停地换进换出操作。而 BOM 算法采用位矩阵进行存储，不需要在内存中存储所有的候选项集，不需要借助频繁(k−1)-项集而直接获得频繁 k-项集，大大地节省了存储空间，具有良好的空间特性。

3.3　总体效果分析

在实际应用中，大型交易事务数据库的使用是非常普遍的，并且大型交易事务数据库中的记录数量也是非常巨大的，

(下转第 197 页)

提取方法，从高阶上抽取了人脸的代数特征，使得抽取得到的各个特征在统计上是独立的，改进了 PCA 算法中只得到不相关特征的弱点。CLDA 方法的加入，使得抽取得到的特征保留了有助于分类的信息，并且本文方法中又利用零空间的信息，得到了更多的分类信息，消除了由于光照、表情等对于人脸分类带来的影响。图 3 和图 4 分别显示不同的人脸识别方法在 ORL 和 Yale 人脸数据库中 10 次不同实验下的识别性能比较，其中的训练样本集由每类人脸的 6 个样本随机组成。从识别性能比较图中可以看出，该方法的识别率始终优于传统的人脸识别方法。由此说明在 ICA 方法中加入了能得到分类信息的 LDA 方法之后，人脸的识别性能有了明显的提高，并且该方法的鲁棒性较好。

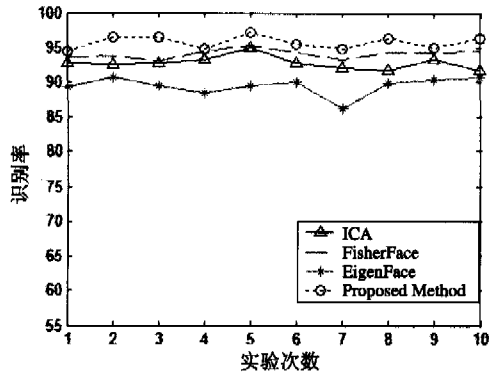

图 3　ORL 人脸数据中不同训练样本集下的识别率比较

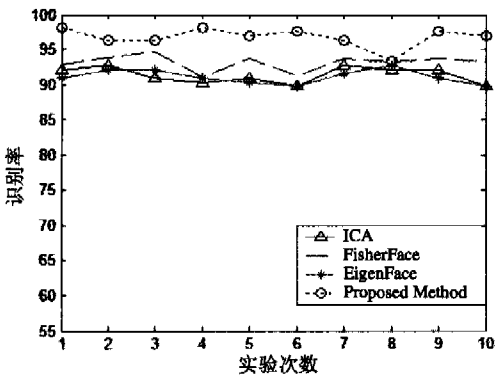

图 4　Yale 人脸数据中不同训练样本集下的识别率比较

结论　本文在 ICA 的基础上，提出了一种基于零空间概念的 LDA 方法与 ICA 方法相结合的算法。通过 ICA 算法可得到统计独立的特征，为了从这个特征向量中得到更多的有助于分类的信息，结合 LDA 方法以得到保留了更多分类信息的特征数据，并且引入零空间的概念，抽取得到经过 ICA 变换之后的所有的分类信息。从而从数据的高级特性中得到了更多有助于分类的特征信息。在 ORL 和 Yale 人脸数据库的实验表明 ICA+LDA 的方法识别率优于 ICA 方法的识别率。

参 考 文 献

1　Juten C, Herault J. Blind separation of source[J], part 1: An adaptive algorithm base on neuromimetic architecture. Signal Processing, 1991, 24(1): 1～10

2　Comon P. Independent component analysis—A new concept? [J]. Signal Processing, 1994, 36(3): 287～314

3　Bartlett M S, Movellan J R, Sejnowski T J. Face Recognition by Independent Component Analysis[J]. IEEE Transaction on Neural Network, 2002, 13(6): 1450～1464

4　Stewart B M, Sejnowski T J. Independent components of face images: A representation for face recognition[A]. In: Proc. of the 4th Annual Joint Symposium on Neural Computation, Pasadena, CA, May 1997

5　Hyvarinen A. Fast and robust fixed-point algorithm for independent component analysis [J]. IEEE Transaction on Neural Network, 1999, 10(3): 626～634

6　杨竹青，李勇，胡德文. 独立成分分析方法综述[J]. 自动化学报, 2002, 28(5): 762～772

7　Belhumeur P N, Hespanha J P, Kriegman D J. Eigenfaces vs Fisherfaces: Recognition Using Class Specific Linear Projection [J]. IEEE Trans. Pattern Analysis and Machine Intelligence, 1997, 19(7): 711～720

8　Yang J, Yang J Y. Why can LDA be performed in PCA transformed space? [J]. Pattern Recognition, 2003, 36(2): 563～566

9　Liu C J, Wechsler H. Comparative assessment of independents component analysis(ICA) for face recognition[A]. In: Second Int. Conf. Audio and Video-based Biometric person Authentication, 1999. 22～24

10　Yi J, Kim J, Choi J, Han J, Lee E. Face Recognition Based on ICA Combined with FLD[A]. Biometric Authentication, 2002. 10～18

11　Yu H, Weng J. Using Discriminant Eigenfeatures for Image Retrieval[J]. IEEE Trans. Pattern Analysis and Machine Intelligence, 1996, 18(8): 831～836

12　Chen L F, Liao H Y M, Lin J C, Ko M T, Yj G J. A New LDA-based Face Recognition System Which Can Solve the Small Sample Size Problem[J]. Pattern Recognition, 2000, 33(10): 1713～1726

13　Yang J, Frangi A F, Yang J Y, Zhang D, Jin Z. KPCA plus LDA: A Complete Kernel Fisher Discriminant Framework for Feature extraction and Recognition. IEEE Trans. Pattern Analysis and Machine Intelligence, 2005, 27(2): 230～244

14　Hyvarinen A, OJA E. Independent component analysis: algorithms and applications [J]. IEEE Trans on Neural Network, 2000, 13(4-5): 411～430

（上接第 189 页）

与 Apriori 算法相比，BOM 算法利用位矩阵可以更方便地来处理数据，矩阵的编码方式也被用来判断一个待生成的 k-项集是不是频繁项集，大大地提高了挖掘关联规则的效率，并且基于矩阵的算法也便于在实际应用中被实现和理解。

结论　为了挖掘大型事务数据库中数据间所有的关联规则，文章提出了一个新的算法 BOM 算法，并且与经典的挖掘关联规则算法 Apriori 算法做了分析对比。BOM 算法具有良好的性能，它不需要多次扫描事务数据库；不需要在内存中存储大量的候选项集，相反，对于一个新的待生成的 k-项集，会立即被判断是否是一个频繁项集，如果不是，则不生成该 k-项集。因此，与 Apriori 算法相比，对于大型交易事务数据库，算法节约了大量的时间和空间，具有良好的性能。

在以后的研究中，将在以下几个方面开展：对如何高效地获得频繁项集的大小 k 进行研究寻找一种更好的数据结构来替代矩阵，进一步降低对内存的需求，提高算法的性能。频繁项集出现的位置并没有给予太多考虑，但是对于一些应用，这将是非常有用的，这也是进一步研究的课题。

参 考 文 献

1　Agrawal R, Srikant R. Mining sequential patterns: [IBM Research Report]. 1995

2　Agrawal R, Imielinski T, Swami A. Mining association rules between sets in large databases. In: Proc. the ACM SIGMOD Conf. Management of Data, May 1993. 207～216

3　Agrawal R, Srikant R. Fast algorithm for mining association rules: [IBM Research Reprot]. 1994

4　Agrawal R, Mannila H, Toivonen H, et al. Fast Discovery of Association Rules. In Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, 1996. 306～328

5　Huang Liusheng, Chen Huaping, Wang Xun, Cheng Guoliang, A Fast Algorithm for Mining Association Rules. In J. Comput. Sci. & Technol., 2000, 15(6): 619～624

6　Jiawei Han, Micheline Kamber. Data Mining: Concepts and Techniques[C]. Mongan Kaufmann publishers, 2000. 225～278