关联规则发现中的聚类方法*)

谢坤武　陈世强　毕晓玲
(湖北民族学院信息工程学院　恩施 445000)

摘　要　算法 MARC(Mining Association Rules using Clustering)将聚类技术应用到关联规则的发现上,MARC 利用聚类技术压缩交易数据库,从而减少开采算法需要处理的数据量以提高开采效率,同时算法提出了聚类汇总转换的概念用以减轻压缩数据带来的信息丢失。在几个实际数据集上的实验表明该算法可以达到高精度和高性能。

关键词　数据开采,聚类分析,关联规则

Clustering Method for Mining Association Rules

XIE Kun-Wu　CHEN Shi-Qiang　BI Xiao-Ling
(School of Information Engineering, Hubei Institute for Nationalities, Enshi 445000)

Abstract　MARC algorithms are proposed to apply clustering analysis to other fields. It integrates clustering into association rules discovery to reduce the size of data sets. It also uses CS (Clustering Summary) transformation to alleviate the loss of information brought by the compression. MARC only needs to scan the database one time. The experiments with several real data sets have demonstrated that MARC can achieve quite well precision and high performance.

Keywords　Data mining, Clustering analysis, Association rules

关联规则发现也是数据开采的一个非常重要的子领域,最初由 Rakesh Agrawal 等人在 1993 年提出[1],用来开采数据间隐含的联系。给定一个交易数据库,数据库中的每一条记录都是一条交易,所谓交易就是由一组项目构成的集合,关联规则用来表达交易中项目之间的联系,例如:"购买啤酒的用户中有 30%也购买了尿布,而在所有的顾客中有 2%同时购买了这两样东西",在这条规则中,30%称作规则的置信度(confidence)、2%称作规则的支持度(support),关联规则开采的目的就是要找出数据集合中满足最小支持度(minsup)和最小置信度(minconf)的所有规则,这里 minsup 和 minconf 是用户预先定义的阈值。

大容量的数据集合能够保证开采出质量较高的关联规则,但是算法效率与数据集合的规模成反比,数据集合规模越大,则算法效率越低。大多数关联规则算法都需要多次扫描数据库,而频繁的 I/O 操作对于大容量数据集在时间耗费上是不能容忍的。为解决以上问题,本文提出算法 MARC (Mining Association Rules using Clustering),它将聚类技术运用到关联规则发现领域,该算法只需对数据库扫描一遍,MARC 算法首先对交易数据库聚类,将类似的交易聚集到同一个类中,然后在聚类结果上而不是在原始数据集上发现关联规则,这样大大减少了关联规则开采算法需要处理的数据量,从而提高开采效率。

1　预备知识

令 I 为 m 个不同项目的集合,即 $I=\{i_1, i_2, \cdots, i_m\}$,$D$ 为交易数据集合,每条交易 T 是一组项目的集合,$T\subseteq I$。一条关联规则具有以下形式:$X\Rightarrow Y$,其中 X, $Y\subset I$,且 $X\cap Y=\emptyset$。项目的集合称作项目集(itemset),项目集的长度为项目集中所包含项目的数量,用 k-itemsets 表示长度为 k 的项目集,k 也称作维数,每个项目集都有一个支持度(support),对于项目集 X,它的支持度记作 sup(X)。给定项目集 $X\subset I$,如果在 D 中包含 X 的交易占整个集合中交易数量的百分比为 s,则 sup(X) = s。规则 $X\Rightarrow Y$ 的支持度定义为 sup($X\cup Y$),置信度定义为 sup($X\cup Y$)/sup(X),那么关联规则开采问题可以描述为从数据集合中生成满足下列两个条件的规则:① 规则的支持度大于给定的阈值最小支持度 minsup;② 规则的置信度大于给定的阈值最小置信度 minconf。满足条件①的项目集称作频繁项目集,否则称作非频繁项目集。

MARC 采用的聚类方法同 CACD 算法[2]基本相同,聚类的形成通过构造一个树形数据结构得以完成,逐个读入数据对象加入到树中,因此构造过程是一个动态过程。这里简单说明对象加入的过程,每读入一个新的数据对象,从树的根节点开始,每到达一层,选择下一层的某一个节点作为新对象的目标节点,直到到达某个叶子节点为止。树的每个叶子节点就是聚类结果,因此聚类是在树形结构构造的过程中逐渐形成的。MARC 和 CACD 使用的树形结构称作聚类树 CT (Clustering Tree),构造聚类树使用了最佳优先搜索策略(best-first search)的思想,当对象到达某个非叶子节点,就面临选择下一个目标节点的问题,算法使用估价函数(evaluation function)作为标准从对象所在子树的下一层中选择最有"希望的(promise)"节点,算法使用对象和节点之间的相似度作为估价函数,记作 sim(A, B),其中 A 和 B 为节点或者对象。假定对象 A 到达节点 B,C_1, C_2, …, C_k 为 B 的子节点,如果 C_i($1\leqslant i\leqslant k$)满足 sim(A, C_i)大于所有 sim(A, C_j)(j = 1, 2, …, k, $C_i\neq C_j$),则 C_i 为最有"希望的"子节点,对象 A 选择 C_i 为下一个到达的节点。

*)国家科技攻关计划项目(编号:2002BA901A02);湖北省科技攻关项目(编号:2004AA210B01)。**谢坤武**　副教授,硕士,主要研究方向为数据挖掘、知识发现;**陈世强**　硕士,讲师,主要研究方向为组件技术及软件开发技术;**毕晓玲**　教授,主要研究方向为数据库及信息处理。

MARC算法首先聚类原始数据库，采用压缩技术使得得到的类表达为压缩形式，从而减少关联规则开采算法需要处理的数据量，算法使用聚类汇总 CS(Clustering Summary)作为类 C 的压缩表达，有关聚类汇总 CS 及其相关定义见文[2]，定义类 C 中所有项目出现的总数量为 $Q_C=\sum_{j=1}^{u} CS.S_j$。

MARC算法步骤如下：① 交易聚类：读入交易对象，逐个插入到聚类树 CT 中，聚类树 CT 存储的数据为压缩后得到的聚类汇总 CS，因此得到的聚类结果数据量比原始集合小得多。该步骤的输出结果为具有压缩特征的聚类集合$\{CS_1, CS_2, \cdots, CS_n\}$。② 将每个聚类汇总 CS_i(i=1, 2,…, n)转换成符合关联规则发现要求的形式 MCS_i，输出结果为$\{MCS_1, MCS_2, \cdots, MCS_n\}$。③ 在步骤②的输出结果上开采关联规则。

2 相关术语

2.1 相似度判别

MARC使用的相似度判别标准是 Jaccard 相似系数在压缩数据 CS 上的扩展，给定交易 T_1 和 T_2，它们之间的相似度使用 Jaccard 相似系数为 $|T_1\cap T_2|/|T_1\cup T_2|$。由于在聚类树的构造中只需比较一个交易 T 和类 C 之间的相似度，因此，MARC只给出交易和类之间的相似度定义。

给定交易 T 和类 C，类 C 的聚类汇总为 $CS(N, I, S)$，且 $|CS.I|=|CS.S|=u$，T 和 C 共有的项目集合定义为

$$M_{TC}=T\cap CS.I \tag{1}$$

M_{TC} 中的项目在 C 中出现的次数总和定义为：

$$n_{MC}=\sum_{j=1}^{u} z_j,\ \text{where } z_j=\begin{cases}0 & \text{if } i_j\notin M_{TC}\\ s_j & \text{if } i_j\in M_{TC}\end{cases},\ \text{where } i_j\in CS.I,\ s_j\in CS.S \tag{2}$$

定义 M_{TC} 中的项目在 T 中出现的次数总和记作 n_{MT}，它实际等于集合 M_{TC} 的势：

$$n_{MT}=|M_{TC}| \tag{3}$$

不属于 M_{TC} 的项目在 C 中出现的次数总和定义为：

$$n_{\overline{M}C}=Q_C-n_{MC} \tag{4}$$

不属于 M_{TC} 的项目在 T 中出现的次数总和定义为：

$$n_{\overline{M}T}=|T|-n_{MT} \tag{5}$$

交易 T 和类 C 的相似度定义为：

$$\begin{aligned}\mathrm{sim}(T, C)&=\frac{n_{MC}+n_{MT}-n_{\overline{M}C}-n_{\overline{M}T}}{|T|+Q_C}\\&=\frac{2(n_{MC}+n_{MT})-|T|-Q_C}{|T|+Q_C}\end{aligned} \tag{6}$$

公式(6)中的分母 $|T|+Q_C$ 使得相似度的值落在[−1, 1]之间，将之规格化，即令相似度的值在[0, 1]之间，得到公式如下：

$$\mathrm{sim}_n(T, C)=\frac{n_{MC}+n_{MT}}{|T|+Q_C} \tag{7}$$

从公式(7)得到交易 T 和类 C 的相似度取决于 $n_{MC}+n_{MT}$，$n_{MC}+n_{MT}$ 的值越大，则 T 和 C 越相似。当类 C 仅包含一条交易时，将类 C 记作 T'，则公式(7)为 $2|T\cap T'|/(|T|+|T'|)$，与 Jaccard 相似系数类似。

给定交易 T 和两个类 C_1、C_2，如果 $\mathrm{sim}_n(T, C_1)\geqslant\mathrm{sim}_n(T, C_2)$，则认为与 C_2 相比，C_1 和 T 之间更相似。举个例子，给定交易 $T=\{a, b\}$，类 C_1 的聚类汇总 $CS_1=(3,\{a, b, c\},\{3, 3, 1\})$，类 C_2 的聚类汇总 $CS_2=(2,\{a, b, c\},\{2, 2, 1\})$，得到 $\mathrm{sim}_n(T, C_1)=(6+2)/(2+7)=0.889$，$\mathrm{sim}_n(T, C_2)=(4+2)/(2+5)=0.857$，因此，与 C_1 相比，C_2 和 T 之间更相似。

2.2 聚类汇总转换

在MARC中，聚类过程同 CACD 相同，只是在构造聚类树过程中比较交易和类的相似度时，采用的计算方法不同。当聚类完成，得到的结果是聚类树中的叶子节点，而聚类树中的每个节点都存储了一个类的压缩形式——聚类汇总 CS，可以将聚类结果记作 $SCS=\{CS_1, CS_2, \cdots, CS_N\}$，其中 N 为聚类树中叶子节点的数目。由于 SCS 中包含的信息集中在项目上，缺少足够的交易信息(只有 $CS.n$ 记录了聚类中交易的数量)，因此 SCS 不能直接用于关联规则发现。$MARC$ 提出需要先转换 SCS 以满足关联规则发现的需求，用 $MSCS$ 表达转换后的 SCS，即 $MSCS=\{MCS_1, MCS_2, \cdots, MCS_N\}$，其中 $MCS_i(i=1, 2, \cdots, N)$ 为转换后的 $CS_i(i=1, 2, \cdots, N)$，转换方法如下：

给定叶子节点 C，它记录的聚类汇总为 CS，$|CS.I|=|CS.S|=u$，MCS 用一个二元组(T, M)表示，其中

$$T=\{\, i \mid i\in CS.I,\ CS.\mathrm{num}(i)\geqslant CS.N*\delta\} \tag{8}$$

$$M=\sum_{j=1}^{|T|} CS.\mathrm{num}(i_j)/|T|,\ \text{where } i_j\in T \tag{9}$$

公式(8)中的 δ 为转换因子，其值域为[0, 1]。二元组中的 M 实际上是 CS 代表的类 C 中所有项目出现次数的平均值。给定某个转换后的 CS 为 $MCS=(T, M)$，将它的两个成员 T 和 M 分别记作 $MCS.T$ 和 $MCS.M$。

举例，给定 $CS=(12,\{a, b, c, d, e, f\},\{3, 5, 1, 9, 7, 11\})$，$\delta=0.25$，那么 $CS.N*\delta=3$，得到 $MCS.T=\{a, b, d, e, f\}$，$MCS.M=(3+5+9+7+11)/5=7$；那么 CS 转换后得到的 $MCS=(\{a, b, d, e, f\}, 7)$。

对于 $MSCS$ 中的每个元素 MCS，可以将 $MCS.T$ 看成项目集或者一条交易的近似值，那么每个 MCS 可以看作是一个近似类，这个近似类中包含了 $MCS.M$ 个相同的交易，交易的形式为 $MCS.T$，举个例子，给定一个包含三条交易的类 C，$T_1=\{a, b\}$，$T_2=\{a, b, c\}$，$T_3=\{a, b, d\}$，容易得到类 C 的 $CS=(3,\{a, b, c, d\},\{3, 3, 1, 1\})$，假设 $\delta=0.5$，那么转换 CS 后得到 $MCS=(\{a, b\}, 3)$，MCS 可以看作包含 3 条相同交易$\{a, b\}$的集合，扩展后形式为：$\{\{a, b\},\{a, b\},\{a, b\}\}$，$MARC$ 算法将 MCS 作为原交易集合 $C=\{\{a, b\},\{a, b, c\},\{a, b, d\}\}$ 的近似集合，因此，$MSCS$ 也可以作为原始数据集合的近似集合进行关联规则的开采，由于这个近似集合仍然是用压缩形式来表达，在其上运行关联规则发现算法效率也会相当高，另一方面，从转换方法看出，压缩形式是通过删除 CS 中的非频繁项目得到的，而这样的操作可能会导致原数据集合中频繁项目集的丢失，不过，从实验中获取的信息表明这种丢失较少发生，同时实验结果也表明这种转换对关联规则开采的结果影响不大。

2.3 支持度和置信度

由于是在压缩数据集合上开采关联规则，因此需要重新定义支持度和置信度，这种重新定义只是根据数据表达形式修改了在 Apriori[3] 算法中的有关定义，内涵还是同经典关联规则的支持度和置信度相同。

项目集的支持度：令$\{MCS_1, MCS_2, \cdots, MCS_N\}$为 N 个 MCS 的集合，$MCS_i=(T_i, M_i)$，$i=1, 2, \cdots, N$。给定项目集 X，X 的支持度记作 $\sup(X)$

$sup(X) = \sum_{i=1}^{N} num_i / \sum_{i=1}^{N} M_i$；其中（$num_i = \begin{cases} 0 & \text{if } X \not\subset T_i \\ M_i & \text{if } X \subseteq T_i \end{cases}$）(10)

上文提到，可以将 *MSCS* 看作原交易集合的近似值，那么项目集 X 的支持度就为包含项目集 X 的交易在 *MSCS* 中所占的百分比。例如，给定 $MSCS = \{(\{a, b\}, 2), (\{a, b, d\}, 2), (\{a, c\}, 1)\}$，将 *MSCS* 看作交易集合$\{\{a, b\}, \{a, b\}, \{a, b, d\}, \{a, b, d\}, \{a, c\}\}$，对于项目集 $X = \{a, b\}$，支持度 $sup(X) = 4 / 5 = 0.8$。

关联规则的支持度和置信度：给定关联规则 $X \Rightarrow Y$，其中 X 和 Y 为项目集。与 Apriori 算法类似，关联规则 $X \Rightarrow Y$ 的支持度和置信度分别为 $sup(X \cup Y)$ 和 $sup(X \cup Y) / sup(X)$。

3 MARC 算法

给定包含 n 条交易的集合 $D = \{T_1, T_2, \cdots, T_n\}$，构造聚类树需要预先定义的阈值 B 和最小相似度 minsim，以及转换 *CS* 为 *MCS* 所需的转换因子 δ，MARC 算法框架描述如下。

```
MARC(D, B, minsim, δ)
i=1,MSCS={},CT = NULL;// 初始化变量
Clustering(D, B, minsim);  // 聚类
for each leaf node CS in CT do  // 转换 SCS 为 MSCS
MSCS = MSCS∪ModifyCS(CS,δ);
GenerateAssociationRules(MSCS);  // 关联规则发现
```

MARC 算法可以划分为三个步骤① 聚类过程、② 转换 SCS 为 MSCS 以及③ 关联规则开采。聚类过程的函数 Clustering(D, B, minsim)同文[2]描述的 CACD 算法基本相同，所以这里不再给出 Clustering 函数的具体描述，但需要说明它与 CACD 不同的地方：① MARC 的聚类过程在聚类树构造完成后，直接将叶子节点作为聚类结果，不需要对它们重新检验，这是因为 MARC 的聚类目的是为了压缩供关联规则发现算法使用的数据集，它对聚类结果的精确度要求并不高；② 由于两个聚类树构造过程中的对象或交易和节点（子类）的相似度定义不同，因此相似度计算的算法不同。

第二步转换过程的主要任务是转换聚类树 CT 中每个叶子节点所记录的 *CS* 为 *MCS*，函数 ModifyCS(*CS*,δ)实现每个 *CS* 的转换，将结果 *MCS* 存放在 MSCS 中。最后一步是在 MSCS 上开采关联规则，函数 GenerateAssociationRules(MSCS)可以采用以往的任意一个关联规则开采算法，如 Apriori，本文不再给出关联规则开采的算法描述。

```
ModifyCS(CS,)
num = CS.N * δ,MCS = {};
for j = 1 to |CS.I| do {
   if CS.num(i_j)≥num then {
      MCS.T = MCS.T∪{CS.I.i_j};
      if CS.num(i_j)>MCS.M then
         MCS.M = CS.num(i_j);
   }
}
return MCS;
```

4 实验分析

4.1 准确度

为比较各种关联规则发现算法生成结果的准确性，这里分别给出频繁项目集和关联规则的相对误差函数用来判断算法的有效性。

一旦给定交易集合 D 和阈值最小支持度 minsup，就可以确定集合 D 中频繁项目集的唯一解，这个解称作标准解，记作 $F = \{f_1, f_2, \cdots, f_u\}$，$u$ 为频繁项目集的个数，$f_i (1 \leqslant i \leqslant u)$ 的支持度记作 $\sup(f_i)$。然后给定训练算法 A，A 在 D 上开采出来的频繁项目集称作 A 的近似解，这个解记作 $F_A = \{f_{A1}, f_{A2}, \cdots, f_{Av}\}$，类似地，$f_{Ai} (1 \leqslant i \leqslant v)$ 的支持度记作 $\sup(f_{Ai})$。令 $F' = F \cup F_A = \{f'_1, f'_2, \cdots, f'_w\}$，其中 $w = |F \cup F_A|$，$f'_j \in F' (1 \leqslant j \leqslant w)$，$f'_j$ 的标准支持度定义为：

$$\sup(f'_j) = \begin{cases} \sup(f_i) & \text{if } f'_j \in F \text{ and } f'_j = f_i, \text{where } 1 \leqslant i \leqslant u \\ 0 & \text{if } f'_j \notin F \end{cases} \quad (11)$$

f'_j 的近似支持度定义为：

$$\sup(f'_{Aj}) = \begin{cases} \sup(f_{Ai}) & \text{if } f'_j \in F_A \text{ and } f'_j = f_{Ai}, \text{where } 1 \leqslant i \leqslant v \\ 0 & \text{if } f'_j \notin F_A \end{cases} \quad (12)$$

那么算法 A 生成的频繁项目集的相对误差 $e_F(A)$ 定义如下：

$$e_F(A) = (\sum_{i=1}^{w} e_i) / w \quad (13)$$

公式(13)中，如果 $\sup(f'_i) > 0$，则 $e_i = |\sup(f'_i) - \sup(f'_{Ai})| / \sup(f'_i)$，否则 $e_i = 1$。

相对误差函数 $e_F(A)$ 是一个判别标准，用来度量算法生成的频繁项目集与标准频繁项目集之间究竟有多接近。从以上定义可明显看出，$e_F(A)$ 的值越小，算法生成的频繁项目集越准确。相对误差 $e_F(A)$ 的值域为 $[0, 1]$，$e_F(A) = 0$ 表示由算法 A 生成的频繁项目集和标准集完全相同。

给定交易集合 D，阈值最小支持度 minsup 和最小置信度 minconf，可以确定集合 D 上的关联规则唯一解 R，这个解也称为标准解。然后给定训练算法 A，可以得到由 A 生成的关联规则 R_T。由算法 A 生成的关联规则的相对误差 $e_R(A)$ 定义为：

$$e_R(A) = 1 - |R \cap R_T| / |R \cup R_T| \quad (14)$$

公式(14)中 $|R \cup R_T| > 0$。与 $e_F(A)$ 类似，$e_R(A)$ 也是一个判别函数，用来衡量算法生成的关联规则的准确率，$e_R(A)$ 的值越小，则算法生成的结果越好。关联规则相对误差 $e_R(A)$ 的值域为 $[0, 1]$，$e_R(A) = 0$ 表示由算法 A 生成的关联规则和标准关联规则完全相同。

假定有算法 A 和算法 B 都用来发现关联规则，给定相同的阈值最小支持度 minsup 和最小置信度 minconf，如果 $e_F(A) \leqslant e_F(B)$ 并且 $e_R(A) \leqslant e_R(B)$，则认为算法 A 得到的结果比算法 B 得到的结果要好。

4.2 实验组织

分别在运行时间和结果误差两方面比较 MARC 算法、Apriori 算法、Sampling 算法[4]和 BIRCH 算法。为描述方便，文中某些地方会将 Apriori、BIRCH、MARC 和 Sampling 算法表示为 A、B、M 和 S。

census 数据集合可以从 http://augustus.csscr.washington.edu/census/comp_013.html 获得，获取的原始数据只抽取人口调查部分作为实验用途，数据库中的数字属性的数据根据文[5]的方法离散化，census 数据包含 49,406 条交易，每条交易有 35 个项目。其它两个数据集合均从 http://www.ics.uci.edu/~mlearn/MLRepository.html 获得，其中一个是蘑菇数据库 mushroom，另一个是 Anonymous Microsoft Web 数据库，为方便起见，简称为 Web 数据库，其中记录了 www.microsoft.com 主页的匿名访问情况，数据从该站点一个星期的访问记录中随机抽取，数据库里的每一条记录列出了一个

用户在一个连续的时间段内访问该站点的页面，每条记录可以看作是一个交易，Web 数据库包含了 32711 条交易、294 个项目，交易最大长度为 30，最小为 1。

表 1　MARC 算法的参数设置

参数	census 数据库	mushroom 数据库	Web 数据库
minconf	30%	20%	30%
minsim	0.75	0.8	0.75
minsup	10%	1%	10%
δ	10%	1%	10%

参数设置：表 1 给出了 MARC 算法在不同数据集合上的参数设置，表中给定的阈值 minconf 和 minsup 同样适用于算法 BIRCH 和 Sampling，除非特别说明，实验所使用的参数按照表 1 取值。

MARC 选择算法 Apriori 作为关联规则开采步骤的算法，另外，由于 BIRCH 是一个聚类算法，为方便比较，实验同样使用 Apriori 算法在 BIRCH 的聚类结果上进行关联规则开采，BIRCH 算法的参数页面大小 P 设置为 1024。Sampling 算法是一个关联规则开采算法，它使用随机取样的方法在样本数据集合生成关联规则，然后在整个数据集合范围内验证局部生成的关联规则，将样本数据集合占整个数据集合的百分比记作 $x\%$，实验中 x 的取值如下，对于 census 和 Web 数据库，$x = 20$，对于 mushroom 数据库，$x=50$。

4.3　算法比较

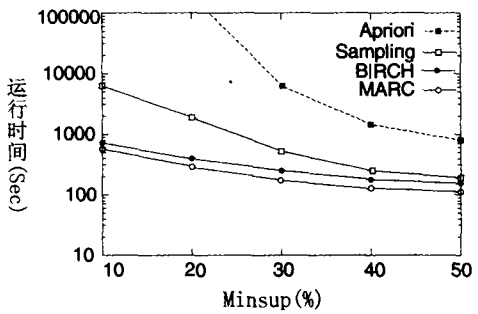

图 1　census 数据库上的运行时间

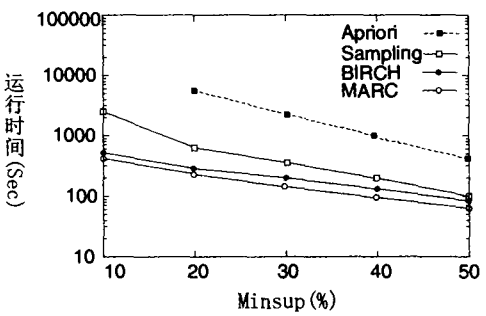

图 2　Web 数据库上的运行时间

图 1 至图 3 分别在 census、Web 和 mushroom 数据库上比较 MARC、Apriori、Sampling 和 BIRCH 的运行时间，实验通过调整 minsup 观察运行时间的变化。在这个实验中，令 MARC 的参数 δ 的值等于 minsup。从图中可以看出，① MARC 和 BIRCH 所需的运行时间均少于 Apriori 和 Sampling，minsup 越低，所耗费时间的差距越明显，其原因是 MARC 和 BIRCH 都在压缩数据集合上发现关联规则；② 单独比较 MARC 和 BIRCH，MARC 的运行时间要小于 BIRCH，这是因为在聚类树构造过程中，MARC 无需重建聚类树，而 BIRCH 要根据内存需要重建 CF 树。

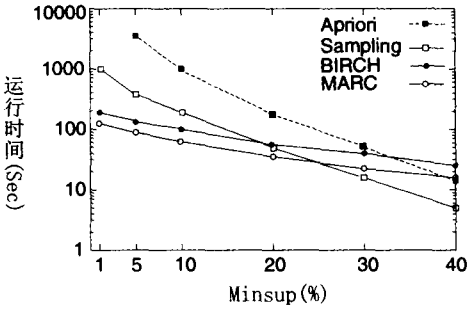

图 3　mushroom 数据库上的运行时间

另外，观察图 3，对于 mushroom 数据库，minsup 较高时，MARC 和 BIRCH 的时间耗费比 Apriori 和 Sampling 高，原因如下，MARC 算法和 BIRCH 在开采关联规则时的时间开销主要由两部分组成：聚类过程和关联规则开采过程，其中聚类过程的时间开销和参数 minsup 无关，因为聚类算法没有用到这个参数，而关联规则开采过程所耗费的时间是和 minsup 的值成反比的，minsup 越大，时间开销越小，这样当 minsup 高过某个门槛值时，Apriori 和 Sampling 这类关联规则算法的运行时间就会快过 MARC 和 BIRCH 算法，对于 mushroom 数据库这样的小型数据集合，这个门槛值不会太高，所以出现了如图 3 所示的情况，但是对于大容量数据库，只有 minsup 的取值很大的时候，Apriori 和 Sampling 才有可能快过 MARC 和 BIRCH，而较高的 minsup 取值使得开采出来的关联规则非常少，因此意义不大。

关联规则开采结果的比较分别在频繁项目集和关联规则上进行。由于 Apriori 算法在完整数据集合上进行，且得到的结果是完全解，由它发现的频繁项目集和关联规则没有误差，因此实验没有计算 Apriori 的结果误差。

表 2 至表 4 分别给出了在不同的 minsup 设置下，三个算法分别在三个数据集上发现的关联规则和频繁项目集的误差率，表 5 则给出了最小置信度不同时，三个算法分别在三个不同数据集合上发现的关联规则的误差率。从这四个表可以得出，MARC 生成的频繁项目集和关联规则的误差率均低于 BIRCH 或 Sampling 算法生成结果的误差率。

表 2　census 数据库的误差率

minsup	BIRCH		MARC		Sampling	
	$e_F(B)$	$e_R(B)$	$e_F(M)$	$e_R(M)$	$e_F(S)$	$e_R(S)$
10%	0.412	0.511	0.014	0.006	0.036	0.027
20%	0.236	0.178	0.012	0.005	0.038	0.028
30%	0.378	0.211	0.013	0.004	0.038	0.028
40%	0.441	0.497	0.012	0.004	0.042	0.031
50%	0.427	0.398	0.011	0.004	0.048	0.037

表 3　mushroom 数据库的误差率

minsup	BIRCH		MARC		Sampling	
	$e_F(B)$	$e_R(B)$	$e_F(M)$	$e_R(M)$	$e_F(S)$	$e_R(S)$
1%	0.214	0.117	0.015	0.008	0.142	0.168
5%	0.222	0.209	0.013	0.008	0.148	0.173
10%	0.246	0.211	0.012	0.007	0.163	0.192
20%	*0.271*	*0.238*	*0.012*	*0.006*	*0.178*	*0.212*
30%	0.223	0.197	0.011	0.007	0.179	0.214

（下转第 214 页）

根据 Wolpert 的"No Free Lunch"理论[21]，任何一种方法不可能对所有领域的问题都是最佳的，不同的分解方法在诸多特征和性能方面存在差异，在实际应用中要根据问题的具体情况和决策表数据本身的特点选择合适的分解方法，力求在保证决策等价的前提下，达到分解效率和分类质量的平衡。

总结与展望 分解是解决大型决策表数据海量性和复杂性问题的有效手段，本文分析了决策表分解的必要性，提出了评价分解方法的三条标准，对当前存在的几种决策表分解方法，从不同的角度分析其特征并进行比较。现有的决策表分解方法还未达到成熟完善的程度，进一步的研究方向包括建立完善的分解终止判断标准，对于递归分解的方法，目前多是结合后续分析的需要或由专家经验确定分解终止的条件，较为主观，应综合考虑多方面因素提出合理全面的标准。从运行效率和分类质量等方面对分解方法进行改进、建立决策等价性判断标准等问题也值得深入研究。

参 考 文 献

1 Jimenez L O, Landgrebe D A. Supervised Classification in High-Dimensional Space: Geometrical, Statistical, and Asymptotical Properties of Multivariate Data. IEEE Transactions on Systems Man, and Cybernetics—Part C: Applications and Reviews, 1998 (28):39～54

2 Zupan B, Bohanec M, Demsar J, et al. Learning by discovering concept hierarchies. Artificial Intelligence, 1999(109):211～242

3 Starzyk J, Nelson D E, Sturtz K. Reduct generation in information systems. Bulletin of international rough set society, 1998,3: 19～22

4 Kumar A. New Techniques for Data Reduction in a Database System for Knowledge Discovery Applications. Journal of Intelligent Information Systems,1998,10(1): 31～48

5 王清毅,范焱,蔡庆生. 知识的约简研究. 小型微型计算机系统, 2000,21(6):623～627

6 He D W, Strege B, Tolle H, et al. Decomposition in Automatic Generation of Petri Nets for Manufacturing System Control and Scheduling. International Journal of Production Research, 2000, 38(6): 1437～1457

7 Michie D. Problem decomposition and the learning of skills. In: Proceedings of the European Conference on Machine Learning, Springer-Verlag, 1995. 17～31

8 Szczerbicki K E, Park K. A Novel Approach to Decomposition of Design Specifications and Search for Solution. International Journal of Production Research,1991,29(7): 1391～1406

9 Kusiak A. Decomposition in data mining: An industrial case study. IEEE Transactions on Electronics Packaging Manufacturing, 2000,23(4):345～353

10 Zupan B, Bohanec M, Bratko I, et al. A dataset decomposition approach to data mining and machine discovery. In: Heckerman D, ed. Proc. of the Third International Conference on Knowledge Discovery and Data Mining. Irvine, CA: AAAI Press,1997. 299～303

11 Zupan B, Bohanec M, Bratko I, et al. Machine learning by function decomposition. In: Proceedings of the Fourteen International Conference on Machine Learning. Nashville, TN: 1997. 421～429

12 杨善林,刘业政,李亚飞. 基于 Rough Sets 理论的证据获取与合成方法. 管理科学学报,2005,8(5):69～75

13 Pawlak Z. Rough Sets. Theoretical Aspects of Reasoning about Data. Dortrecht: Kluwer Academic Publishers,1991

14 樊群,赵卫东,达庆利. 一种基于粗集的实例分解归纳学习方法. 管理工程学报,2001,15(2):79～81

15 Nguyen S H, Skowron A. Searching for Relational Pattern on Data. In: Komorowski J, Zytkow J, eds. In: Proceedings of First European Symposium on Principles of Data Mining and Knowledge Discovery. Trondheim, Norway: Springer Verlag,1997. 265～276

16 Nguyen S H, Nguyen T T, Polkowski L, et al. Decision rules for large data tables. In: Proceedings of Symposium on Modeling, Analysis and Simulation. France, Lille,1996. 942～947

17 Nguyen S H, Polkowski L, Skowron A, et al. Searching for Approximate Description of Decision Classes. In: Proceedings of the Fourth International Workshop on Rough Sets, Fuzzy Sets and Machine Discovery. Tokyo, Japan,1996. 153～161

18 Nguyen S H, Skowron A, Synak P, et al. Knowledge discovery in data bases: Rough set approach. In: Mares M, ed. Proceedings of the Seventh International Fuzzy Systems Association World Congress. Academia, rague,1997. 204～209

19 马昕,孙优贤. 面向大型数据表的粗分析方法. 计算机工程与应用,2003(16):198～200

20 王庆东,马昕,戴华平,等. 基于粗糙集属性量度的数据库分解方法. 浙江大学学报(工学版),2004,38(9):1196～1199

21 Wolpert D H. The lack of a priori distinctions between learning algorithms. Neural Computation,1996(8): 1341～1390

(上接第 183 页)

表 4 Web 数据库的误差率

minsup	BIRCH		MARC		Sampling	
	$e_F(B)$	$e_R(B)$	$e_F(M)$	$e_R(M)$	$e_F(S)$	$e_R(S)$
10%	0.361	0.412	0.021	0.012	0.031	0.022
20%	0.213	0.127	0.020	0.012	0.032	0.027
30%	0.288	0.256	0.020	0.009	0.033	0.028
40%	0.279	0.301	0.020	0.011	0.036	0.030
50%	0.261	0.204	0.022	0.010	0.037	0.029

表 5 关联规则的误差率

min-conf	Census			mushroom			Web		
	$e_R(B)$	$e_R(M)$	$e_R(S)$	$e_R(B)$	$e_R(M)$	$e_R(S)$	$e_R(B)$	$e_R(M)$	$e_R(S)$
10%	0.378	0.007	0.026	0.366	0.011	0.154	0.362	0.013	0.021
20%	0.391	0.005	0.027	0.198	0.009	0.159	0.471	0.009	0.022
30%	0.511	0.006	0.027	0.117	0.008	0.168	0.412	0.012	0.022
40%	0.414	0.006	0.028	0.293	0.009	0.171	0.402	0.009	0.022
50%	0.376	0.006	0.028	0.312	0.008	0.172	0.365	0.010	0.023

总结 实验结果表明，与 BIRCH 和 Sampling 算法相比，在运行时间和算法结果上综合比较，MARC 优势明显，虽然在运行时间比较上，BIRCH 和 MARC 耗时基本相同，但是由于 BIRCH 只适合开采数字数据，在交易数据上得到的结果并不理想。

参 考 文 献

1 Agrawal R, Imielinski T, Swami A N. Mining Association Rules between Sets of Items in Large Databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data (SIGMOD'93). 1993. 207～216

2 谢坤武,陈世强. 一种分类数据的聚类算法. 见:全国数据库学术会议(NDBC2006). 广州,2006. 332～327

3 Agrawal R, Srikant R. Fast Algorithms for Mining Association Rules in Large Databases. In: Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94). 1994. 487～499

4 Toivonen H. Sampling Large Databases for Association Rules. In: Proceedings of 22nd International Conference on Very Large Data Bases (VLDB'96). 1996. 134～145

5 Brin S, Motwani R, Ullman J D, et al. Dynamic Itemset Counting and Implication Rules for Market Basket Data. In: Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD'97). 1997. 255～264