一种高效的核属性动态更新算法

钱文彬[1] 杨炳儒[1] 徐章艳[1,2] 李 慧[1]
(北京科技大学计算机与通信工程学院 北京 100083)[1]
(广西师范大学计算机科学与信息工程学院 桂林 541004)[2]

摘 要 针对决策表中对象动态删除的情况,研究了核属性的动态更新问题。首先引入了简化决策表的概念,剔除了大量重复冗余的对象,然后详细分析了当决策表删除对象时核属性的动态更新机制,并将逐层细化的方法应用到核属性的动态更新中,避免了许多不必要的重复计算。在此基础上,设计了一种无需存储差别矩阵的核属性动态更新算法。当决策表有对象删除时,该算法只需扫描一遍变化后的决策表,便可快速对核属性进行动态更新。最后,通过实例分析和实验比较验证了算法的可行性和有效性。

关键词 粗糙集理论,核属性,动态更新,决策表,算法复杂度
中图法分类号 TP18 **文献标识码** A

Efficient Dynamic Updating Algorithm of the Computation of Core in Decision Table

QIAN Wen-bin[1] YANG Bing-ru[1] XU Zhang-yan[1,2] LI Hui[1]
(School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China)[1]
(School of Computer Science and Information Engineering, Guangxi Normal University, Guilin 541004, China)[2]

Abstract The dynamic updating algorithm of computation of core was discussed in the decision table. Aiming at this situation, the concept of the simplified decision table was first introduced, and a large number of repeated objects in decision table were deleted effectively, what's more, some dynamic updating mechanisms were analyzed when the objects were deleted in original decision table. At the same time, in order to avoid needless repeated computation, multi-level hierarchical model was applied to dynamic updating process. On this condition, an efficient dynamic updating algorithm for computing core was proposed, which does not store discernibility matrix. The algorithm only scans updated decision table to compute core when the objects are dynamically deleted. At last, theoretical analysis and experimental results show that the algorithm is feasible and effective.

Keywords Rough set theory, Core, Dynamic updating, Decision table, Algorithm complexity

1 引言

粗糙集理论是波兰科学家 Pawlak 教授于 1982 年提出的一种能有效处理不精确、不完备知识的数学理论[1,2],现已在数据挖掘、知识发现、模式分类、机器学习、智能决策等方面得到广泛应用。求解核属性和属性约简是粗糙集理论研究的主要内容。为此,如何快速求解决策表的核属性,具有重要的研究意义。

近年来,学者根据不同的需求,设计了许多不同的求核算法[3-9],算法的计算效率也取得了显著进步。差别矩阵方法是求核算法的主要方法之一[3],许多学者对其进行了大量的研究。Hu 等学者构造了决策表的差别矩阵,并设计了该差别矩阵的求核算法[4],但该方法求得的核有时并不正确。为此,叶东毅教授[5]对差别矩阵进行了改进,提出了一种新的求核算法。王国胤教授[6]分析了 Hu 方法所求得的核不正确的原因是它不能处理不一致的决策表。为了处理决策表中不一致对象,杨明教授[7]提出了一种新的修正的差别矩阵及其求核方法,并证明了所求得的核属性是正确的。冯林博士[8]利用 SQL 操作语句提出了一种求解数据库的核属性算法。上述研究都是针对静态决策表而设计的,并不适合动态变化的情况。

在实际应用中,决策表往往是动态变化的,存在对象增加和删除的情况,从而需要对核属性进行动态更新。对于动态变化不大的决策表而言,使用静态算法将会使得原来核属性中许多有用的信息未能得到有效的利用。为此,文献[10-12]中提出了只考虑决策表中对象动态增加时核属性的更新算法。针对决策表中对象动态删除时的更新算法的报道并不多。杨明教授[13]设计了一种基于差别矩阵的核属性快速更

到稿日期:2011-08-12 返修日期:2011-11-23 本文受国家重点基础研究发展计划项目(973 计划)(2009CB522701),国家自然科学基金项目(60963008,60875029),科技部创新方法专项项目(2010IM020900),广西自然科学基金项目(2011GXNSFA018163)资助。

钱文彬(1984-),男,博士生,主要研究方向为粗糙集理论、数据挖掘,E-mail:qianwenbin1027@126.com;杨炳儒(1943-),男,教授,博士生导师,主要研究方向为人工智能、数据挖掘;徐章艳(1972-),男,博士,教授,主要研究方向为粗糙集、模糊集、数据挖掘;李 慧(1983-),女,博士生,主要研究方向为数据挖掘、模糊认知图。

新算法，但算法需预先存储决策表的差别矩阵，求解核时需遍历整个差别矩阵。当面对大规模数据集时，差别矩阵所占用的内存是难以承受的。冯少荣博士[14]提出了一种改进的核增量式更新算法，但该算法未对决策表进行简化，且存在许多不必要的重复计算。当删除对象的数量较大时，算法的计算效率难以提高。

针对现有的动态求核算法计算效率不理想的情况，本文首先将决策表的求核转化到简化决策表上求解；然后详细分析了决策表中出现对象动态删除的情况下核属性的更新变化机制，并将逐层细化的方法应用到核属性的动态更新过程，剔除了许多不必要的重复计算，从而可有效提高核属性的动态更新效率。在此基础上，提出了一种无需存储差别矩阵的核属性动态更新算法，显著地减少了算法所需的空间。理论分析和实验结果表明，本文提出的算法是有效、可行的。

2 相关知识

定义 1(信息系统，决策表) $S=(U,A,V,f)$，其中 $U=\{x_1,x_2,\cdots,x_n\}$ 表示非空有限的对象集合，也称论域；属性集 $A=C\cup D$ 且 $C\cap D=\varnothing$(其中 C 为条件属性集，D 为决策属性集)，$V=\bigcup_{a\in C\cup D}V_a$，其中 V_a 是属性 a 的值域，$f:U\times C\cup D\to V$ 是一个信息函数，即 $\forall a\in C\cup D,x\in U$ 有 $f(x,a)\in V_a$；每一个属性子集 $B\subseteq(C\cup D)$ 决定一个二元不可辨别关系 $IND(B)=\{(x,y)\in U\times U\mid\forall a\in B,f(x,a)=f(y,a)\}$，$IND(B)$ 可简记 U/B。

定义 2 在决策表 $S=(U,C,D,V,f)$ 中，对于 $x,y\in U$，如果 $\forall c_i\in C$ 存在 $f(x,c_i)=f(y,c_i)\wedge f(x,D)=f(y,D)$，则称对象 x 与 y 是不一致的，否则称这两个对象是一致的。包含不一致对象的决策表称为不一致决策表，否则称为一致决策表。

定义 3 在决策表 $S=(U,C,D,V,f)$ 中，对于 $\forall B\subseteq C\cup D$，设 $U/B=\{B_1,B_2,\cdots,B_m\}$，若 $X\subseteq U$，则称 $B\_(X)=\cup\{B_i\mid B_i\in U/B,B_i\subseteq X\}$ 为 X 关于 B 的下近似集。

定义 4 在决策表 $S=(U,C,D,V,f)$ 中，设 $U/D=\{D_1,D_2,\cdots,D_r\}$ 表示论域 U 在决策属性集 D 下的划分，$U/B=\{B_1,B_2,\cdots,B_m\}$ 表示论域 U 在条件属性集 $B(B\subseteq C)$ 下的划分，称 $POS_B(D)=\bigcup_{D_i\in U/B}B\_(D_i)$ 为 B 关于 D 的正区域。

定义 5 在决策表 $S=(U,C,D,V,f)$ 中，对于 $\forall B\subseteq C$，存在 $POS_B(D)=POS_C(D)$，且若 $\forall c_i\in B,POS_{B-\{c_i\}}(D)\neq POS_C(D)$，则称 B 为 C 相对于 D 的基于正区域的属性约简。

定义 6 在决策表 $S=(U,C,D,V,f)$ 中，若 $c_k\in C$，存在 $POS_{C-\{c_k\}}(D)\neq POS_C(D)$，则称属性 c_k 在 C 中是不可省的；记 C 相对于 D 的所有基于正区域的属性约简集为 $Red(C)$，则称 $Core(C)=\cap Red(C)$ 为基于正区域的核属性集。

性质 1[2] 在决策表 $S=(U,C,D,V,f)$ 中，若 $c_k\in Core(C)$，则属性 c_k 是不可省的。

定义 7[5] 设在决策表 $S=(U,C,D,V,f)$ 中，记 $U/C=\{[x_1']_C,[x_2']_C,\cdots,[x_m']_C\}=\{X_1,X_2,\cdots,X_m\}$，记 $U'=\{x_1',x_2',\cdots,x_m'\}$，设 $POS_C(D)=[x_{i_1}']_C\cup[x_{i_2}']_C\cup\cdots[x_{i_t}']_C$，其中 $\{x_{i_1}',x_{i_2}',\cdots,x_{i_t}'\}\subseteq U'$ 且 $|[x_{i_s}']_C/D|=1(s=1,2,\cdots,t)$；记 $U'_{pos}=\{x'_{i_1},x'_{i_2},\cdots,x'_{i_t}\}$，$U'_{neg}=U'-U'_{pos}$；则称 $S'=(U',C,D,V,f)$ 为简化决策表。

定义 8 在决策表 $S=(U,C,D,V,f)$ 中，$S'=(U',C,D,V,f)$ 为简化决策表，简化的差别矩阵定义为 $M'=((m(i',j'),k))$，其元素的定义如下：

$$m((i',j'),k)=\begin{cases}c_k, & c_k\in C,f(x_i',c_k)\neq f(x_j',c_k)\wedge f(x_i',D)\neq f(x_j',D)\\ & \text{且 } x_i',x_j'\text{ 在 }U'_{pos}\text{ 中}\\ c_k, & c_k\in C,f(x_i',c_k)\neq f(x_j',c_k)\text{ 且 }x_i'\text{ 和 }x_j'\text{ 一个在 }U'_{pos},\\ & \text{一个在 }U'_{neg}\text{ 中}\\ \varnothing, & \text{否则}\end{cases}$$

定义 9 设 $M'=((m(i',j'),k))$ 是决策表 $S=(U,C,D,V,f)$ 的简化差别矩阵，记 $SCore(C)=\{c_k\in m(i',j')\wedge|m(i',j')|=1\}$，则称 $SCore(C)$ 为简化决策表的核属性集。

定理 1 在决策表 $S=(U,C,D,V,f)$ 中，$S'=(U',C,D,V,f)$ 为简化决策表，则有 $SCore(C)=Core(C)$ 成立。

证明：任取 $c_k\in SCore(C)$，则存在 x_i' 和 x_j'，使得 $c_k\in m(i',j')\wedge|m(i',j')|=1$。当 x_i' 和 x_j' 都在 U'_{pos} 中时，则 $[x_i']_C\cup[x_j']_C\subseteq[x_i']_{C-\{c_k\}}$(因为 $[x_i']_C$ 和 $[x_j']_C$ 只能根据属性 c_k 才能区分)，而 $f(x_i',D)\neq f(x_j',D)$，故 $([x_i']_C\cup[x_j']_C)\not\subset POS_{C-\{c_k\}}(D)$；一方面，由于 $([x_i']_C\cup[x_j']_C)\subseteq POS_C(D)$，因此 $POS_{C-\{c_k\}}(D)\neq POS_C(D)$，由定义 8 可知 $c_k\in Core(C)$；当 x_i' 和 x_j' 一个在 U'_{pos} 而另一个在 U'_{neg} 中时，无妨设 $x_i'\in U'_{pos}$ 和 $x_j'\in U'_{neg}$，同理可知 $[x_i']_C\cup[x_j']_C\subseteq[x_i']_{C-\{c_k\}}$，且由于 $x_j'\in U'_{neg}$，故 $([x_i']_C\cup[x_j']_C)\not\subset POS_{C-\{c_k\}}(D)$，另一方面，由 $[x_i']_C\subseteq POS_C(D)$，故 $POS_{C-\{c_k\}}(D)\neq POS_C(D)$，由性质 1 知 $c_k\in Core(C)$。由 c_k 的任意性知 $SCore(C)\subseteq Core(C)$。

任取 $c_k\in Core(C)$，则有 $POS_{C-\{c_k\}}(D)\neq POS_C(D)$，则存在 $x_i'\in U'_{pos}$ 使得 $[x_i']_{C-\{c_k\}}\not\subset POS_{C-\{c_k\}}(D)$，从而至少存在 $x_j'\in[x_i']_{C-\{c_k\}}\wedge x_j'\notin[x_i']_C(x_j'\in U')$，使得 $f(x_i',D)\neq f(x_j,D)$。由 $x_j'\notin[x_i']_C$ 知 $f(x_i',a)\neq f(x_j',a)$。由定义 9 有 $c_k\in m(i',j')\wedge|m(i',j')|=1$，从而有 $c_k\in SCore(C)$。由 c_k 的任意性知 $SCore(C)\supseteq Core(C)$。

综上所述，可知 $SCore(C)=Core(C)$ 成立。证毕。

3 核属性的动态更新

3.1 对象动态删除下核属性的更新机制

下面针对决策表中对象动态删除的情况分别进行讨论，并给出与之相对应的核属性动态更新变化机制。若假设要删除的对象为 x，则

1)如果 $x\in X_t\wedge x\in U'_{pos}$，存在 $|X_t|>1$，则核属性集 $Core(C)$ 保持不变。

2)如果 $x\in X_t\wedge x\in U'_{pos}$，存在 $|X_t|=1$，则需计算对象 x 与 $x'\in\{x'\mid(x'\in U'_{pos}\wedge f(x,D)\neq f(x',D))\vee x'\in U'_{neg}\}$ 所对应的矩阵元素；根据核属性定义对核属性进行删除更新。

3)如果 $x\in X_t\wedge x\in U'_{neg}$，存在 $|X_t|>2$ 且 $\exists y,z\in X_t\wedge f(y,D)\neq f(z,D)$ 时，则核属性集 $Core(C)$ 保持不变。

4)如果 $x\in X_t\wedge x\in U'_{neg}$，存在 $|X_t|>2$ 且 $\forall y,z\in X_t\wedge f(y,D)=f(z,D)$，则需计算对象 x 与 $x'\in\{x'\mid x'\in U'_{pos}\wedge$

$f(y,D)=f(x',D)\}$所对应的矩阵元素，根据核属性定义对核属性进行删除更新；同时需计算对象 x 与 $x'\in\{x'\mid x'\in U'_{neg}\}$所对应的矩阵元素，根据核属性定义对核属性进行增加更新。

5)如果 $x\in X_t\wedge x\in U'_{neg}$，存在$|X_t|=2$且 $y\in X_t$，则需计算对象 x 与 $x'\in\{x'\mid x'\in U'_{pos}\wedge f(y,D)=f(x',D)\}$所对应的矩阵元素，根据核属性定义对核属性进行删除更新；同时需计算对象 x 与 $x'\in\{x'\mid x'\in U'_{neg}\}$所对应的矩阵元素，根据核属性定义对核属性进行增加更新。

3.2 算法描述

在动态更新算法中，对每个属性设置核属性的标识位 *coreflag*，每个属性的 *coreflag* 值为该属性在简化决策表中为核属性的次数。例如 $c_k.coreflag=2$，说明属性 c_k 是核属性且存在两组不同的对象产生的核属性是 c_k。

算法 高效的核属性动态更新算法

输入：简化决策表 $S'=(U'_{pos}\cup U'_{neg},C,D,V,f)$，原决策表的核属性 Core(C)，U/C={$X_1,X_2,\cdots,X_m$}，每个属性的 coreflag 值，需要删除对象 $x\in U'$。

输出：更新后的 Core(C)。

```
Begin
Step1  If (x∈U'pos)
  { //Xt. count 表示集合 Xt 中对象的个数
    If  (x∈Xt∧Xt. count>1) // 第①种情况
      Core(C)保持不变;
    else //第②种情况
      {
        需要计算对象 x 与 x'∈{x'|(x'∈U'pos∧f(x,D)≠ f(x',D))
        ∨x'∈U'neg}所对应的差别矩阵 m(i,j);
        Del_Core(m(i,j),Core(C));
        U'pos=U'pos-{x};
        }
      Xt. count=Xt. count-1;
      Xt=Xt-{x};
}
Step2  if(∃y∈U'neg∧f(ci,x)=f(ci,y))
  {//第③种情况
    If (x∈Xt∧Xt. count>2∧∃y,z∈Xt∧f(y,D)≠f(z,D))
       Core(C)保持不变;
    else //第④⑤种情况
    {
      需要计算对象 x 与 x'∈{x'|x'∈U'pos∧f(y,D)==f(x',D)}所
      对应的差别矩阵 m(i,j);
      Del_Core(m(i,j),Core(C));
      需要计算对象 x 与 x'∈{x'|x'∈U'neg}所对应的差别矩阵 m(i,
      j);
      Add_Core(m(i,j),Core(C));
      f(x,D)=f(y,D);
      U'pos=U'pos∪{x};
      U'neg=U'neg-{x};
    }
  Xt. count=Xt. count-1;
  Xt=Xt-{x};
  }
End
```

对于上述算法中的核属性更新判别函数描述如下。

```
//核属性的动态增加
  Add_Core(m(i,j),Core(C))
  {
    if(|m(i,j)|)==1∧ck∉Core(C))
      {Core(C)=Core(C)∪{ck};
        ck. coreflag=ck. coreflag+1;
        }
    else if(|m(i,j)|)==1∧ck∈Core(C))
      {Core(C)保持不变;
      ck. coreflag=ck. coreflag+1;
      }
    else
      Core(C)保持不变;
}
//核属性的动态删除
  Del_Core(m(i,j),Core(C))
  {
    if(|m(i,j)|)==1∧ck. coreflag≥2)
    {Core(C)保持不变;
      ck. coreflag=ck. coreflag-1;
      }
    else if(|m(i,j)|)==1∧ck. coreflag==1)
      {Core(C)=Core(C)-{ck};
      ck. coreflag=ck. coreflag-1;
      }
    else
      Core(C)保持不变;
}
```

算法复杂度分析：

若删除的对象 x 属于核属性动态更新的第①种情况，算法的最坏时间复杂度为 $O(|C|\times|U'_{pos}|)$；若删除的对象 x 属于核属性动态更新的第②种情况，算法的最坏时间复杂度为 $O(|C|\times(|U'_{pos}|+|U'|))$；若删除的对象 x 属于核属性动态更新的第③种情况，算法的最坏时间复杂度为 $O(|C|\times|U'_{neg}|)$；若删除的对象 x 属于核属性动态更新的第④和第⑤种情况，算法的最坏时间复杂度为 $O(|C|\times(|U'_{neg}|+|U'|))$；因此该动态更新算法的时间复杂度为 $O(|C|\times|U'|)$。由于算法无需构建差别矩阵，易知动态更新算法的空间复杂度为 $O(|C|)$。

在文献[13]中算法的时间复杂度和空间复杂度均为 $O(|C|\times|U_1|\times(|U_1|+|U_2'|))$，其中 U_1 表示决策表中未简化的一致对象集，U_2' 表示决策表中简化的不一致对象集。文献[14]中算法的时间复杂度和空间复杂度分别为 $O(|C|\times|U|)$和 $O(|C|)$，因此新算法在效率和可操作性上好于文献[13,14]中的核更新算法，而且新算法无论当决策表中内含不一致对象较多还是一致对象较多时，都能较好地降低算法的时空复杂度，使之能适应大规模决策表的数据处理。

4 实例分析及实验比较

4.1 实例分析

下面以表 1 的决策表为例对新算法进行说明。其中 c_1，

c_2, c_3, c_4 为条件属性，D 为决策属性，共 15 个对象。

首先对决策表进行简化，得到简化决策表 $S'=(U'_{pos} \cup U'_{neg}, C, D, V, f)$，其中 $U'_{pos}=\{x_1, x_3, x_8\}$，$U'_{neg}=\{x_2, x_5, x_{11}\}$，$U/C=\{\{x_1\}, \{x_2, x_4, x_{13}\}, \{x_3, x_6, x_{10}\}, \{x_5, x_7\}, \{x_8, x_9, x_{12}\}, \{x_{11}, x_{14}, x_{15}\}\}$。根据定义 6 可知，核属性集为 $Core(C)=\{c_3, c_4\}$，同时可得 $c_1.coreflag=0, c_2.coreflag=0, c_3.coreflag=2, c_4.coreflag=1$。

下面通过动态删除决策表中的对象来详细说明核属性的更新变化机制，算法有效地实现了对核属性集 $Core(C)$ 的快速更新。

1）当动态删除的对象 $x=(0,1,2,1,0)$ 时，由于 $\exists x_8 \in U'_{pos}$ 且 $x_8 \in X_5 \wedge X_5.count=3$，根据核属性更新变化机制中的第①种情况，可知 $Core(C)$ 保持不变。

2）当动态删除的对象 $x=(1,2,1,1,0)$ 时，由于 $\exists x_1 \in U'_{pos}$ 且 $x_1 \in X_1 \wedge X_1.count=1$，根据核属性更新变化机制中的第②种情况，通过计算可知 $m(1,3)=c_3$ 和 $m(1,5)=c_4$，因此更新后的核属性集为 $Core(C)=\{c_3\}$，并同时更新 $c_3.coreflag=1, c_4.coreflag=0$。

3）当动态删除的对象 $x=(0,2,1,0,0)$ 时，由于 $\exists x_{11} \in U'_{neg}$ 且 $x_{11} \in X_6 \wedge x_{14}, x_{15} \in X_6 \wedge f(x_{14}, D) \neq f(x_{15}, D)$，根据核属性更新变化机制中的第③种情况，可知 $Core(C)$ 保持不变。

4）当动态删除的对象 $x=(0,1,0,0,1)$ 时，由于 $\exists x_2 \in U'_{neg}$ 且 $x_2 \in X_2 \wedge x_4, x_{13} \in X_2 \wedge f(x_4, D)=f(x_{13}, D)$，根据核属性更新变化机制中的第④种情况，通过计算可知 $m(4,8)=c_3$，因此更新后的核属性集为 $Core(C)=\{c_3, c_4\}$，并同时更新 $c_3.coreflag=1$。

5）当动态删除的对象 $x=(1,2,1,0,1)$ 时，由于 $\exists x_5 \in U'_{neg}$ 且 $x_5 \in X_2 \wedge x_7 \in X_2$，根据核属性更新变化机制中的第⑤种情况，通过计算可知 $m(1,7)=c_4$ 和 $m(7,11)=c_1$，因此更新后的核属性集为 $Core(C)=\{c_1, c_3\}$，并同时更新 $c_1.coreflag=1, c_4.coreflag=0$。

表 1　决策表

数据集	条件属性				决策属性
U	c_1	c_2	c_3	c_4	D
x_1	1	2	1	1	0
x_2	0	1	0	0	1
x_3	1	2	0	1	1
x_4	0	1	0	0	0
x_5	1	2	1	0	1
x_6	1	2	0	1	1
x_7	1	2	1	0	0
x_8	0	1	2	1	0
x_9	0	1	2	1	0
x_{10}	1	2	0	1	1
x_{11}	0	2	1	0	0
x_{12}	0	1	2	1	0
x_{13}	0	1	0	0	0
x_{14}	0	2	1	0	1
x_{15}	0	2	1	0	0

4.2　实验比较

为进一步测试算法的性能，从 UCI 机器学习数据库中选取了数据集 mushroom 和 Car 作为实验的测试数据进行实验对比。硬件环境：Intel Core(TM)2，CPU E7400 2.8 GHz 和 2 GB 的内存，在开发平台 Visual Studio 2005 中 C++编程实现。为便于算法的性能比较，将文献[13]中的算法简记为 Y_Algorithm 算法，将文献[14]中的算法简记为 F_Algorithm 算法，将本文的增量更新算法简记为 Q_Algorithm 算法（注：数据集 mushroom 共 8124 个对象、22 个条件属性、1 个决策属性；数据集 Car 共 1728 个对象、6 个条件属性、1 个决策属性）。为此分别设计了以下两组实验：第一组实验为在数据集 mushroom 中分别随机删除 100，200，300，400 个对象；第二组实验为在数据集 Car 中分别随机删除 50，100，150，200 个对象。每个算法执行 10 次，算法的执行时间为 10 次运行时间的平均值，实验结果如图 1 和图 2 所示。

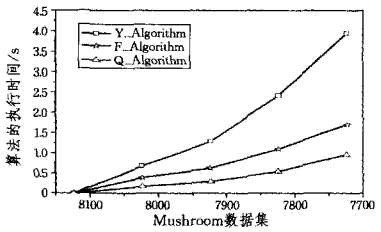

图 1　Mushroom 数据集上的算法性能比较

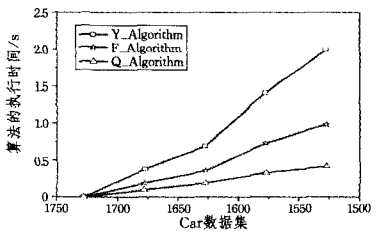

图 2　Car 数据集上的算法性能比较

从图 1 可以看出，Q_Algorithm 算法的计算效率优于 Y_Algorithm 和 F_Algorithm 算法。这是因为数据集 Mushroom 中 8124 个对象是一致的且不能简化，Y_Algorithm 算法需预先存储决策表的差别矩阵，再扫描该差别矩阵的行和列，为此需要耗费大量的计算时间，算法在执行过程中内存占用严重。F_Algorithm 算法对核属性动态更新时，同样存在许多重复的计算，使得算法的效率不够理想。从图 2 中可以看出，Q_Algorithm 算法的效率优越性则更加明显，主要原因是数据集 Car 中 1728 条对象存在不一致性对象和重复对象，Y_Algorithm 算法只简化了决策表中的不一致对象，未考虑决策表中存在重复对象的情况；F_Algorithm 算法未对决策表简化而直接对核属性进行动态更新，Q_Algorithm 算法不但剔除了决策表中的不一致对象，同时剔除了决策表中的重复对象，使得简化后的原决策表对象数为 972，有效地缩小了核属性动态更新时的搜索空间，使得算法效率得到显著的提高。从上述两组实验对比可知，当有对象在决策表动态删除时，Q_Algorithm 算法无需存储差别矩阵，只需判断删除的对象与原决策表中的对象属于哪种情况，然后根据更新变化机制对核属性进行动态更新，从而有效节省了算法的存储空间和计算耗时，因此本文算法是一种相对高效的动态更新算法。

结束语　已有的决策表动态求核算法主要考虑决策表中对象动态增加的情况，对于对象动态删除情况的研究相对较少。为此本文主要考虑了决策表中对象动态删除情况下核属性的更新问题，详细讨论了对象删除时核属性的动态更新机制，在此基础上提出了一种无需存储差别矩阵的核属性动态

更新算法。算法将对原决策表的核属性更新转换为对简化决策表的核属性更新，并且只需扫描一次变化后的决策表便可对核属性进行快速更新，有效地提高了核属性的更新效率。理论分析和实验结果表明，本文算法在效率和可操作性上优于已有的同类算法。求解核属性是属性约简的重要步骤，如何利用更新的核属性对决策表进行动态属性约简，将是下一步的研究重点。

参 考 文 献

[1] Pawlak Z. Rough sets[J]. International Journal of Computer and Information Science, 1982, 11 (5): 341-356

[2] Pawlak Z, Skowron A. Rudiments of rough sets[J]. Information Sciences, 2007, 117(1): 3-37

[3] Hu X H, Cercone N. Learning in relational databases: A rough set approach[J]. Computational Intelligence, 1995, 11(2): 323-337

[4] Skowron A, Rauszer C. The discernibility matrices and functions in information systems[Z]. Intelligent Decision Support Handbook of Applications and Advances of the Rough Sets Theory, Dordrecht: Kluwer Academic Publisher, 1991: 331-362

[5] 叶东毅，陈昭炯. 一个新的差别矩阵及其求核方法[J]. 电子学报，2002，30(7)：1086-1088

[6] 王国胤. 决策表核属性的计算方法[J]. 计算机学报，2003，26(5)：611-615

[7] 杨明，孙志挥. 改进的差别矩阵及其求核方法[J]. 复旦学报：自然科学版，2004，43(5)：865-868

[8] 冯林，李天瑞. 基于 SQL 的属性核与约简高效计算方法[J]. 计算机科学，2010，37(1)：236-238

[9] 徐章艳，刘作鹏，杨炳儒，等. 一个复杂度为 max {O(|C||U|)，O(|C|²|U/C|)}的快速属性约简算法[J]. 计算机学报，2006，29(3)：391-399

[10] 钱文彬，徐章艳，杨炳儒，等. 一种基于决策表的核增量式高效更新算法[J]. 小型微型计算机系统，2010，31(4)：739-743

[11] 杨明. 一种基于改进差别矩阵的核增量式更新算法[J]. 计算机学报，2006，29(3)：407-413

[12] 冯少荣，张东站. 一种高效的增量式属性约简算法[J]. 控制与决策，2011，26(4)：495-500

[13] 杨明，杨萍. 基于差别矩阵的属性核快速更新算法[J]. 控制与决策，2007，22(4)：453-456

[14] 冯少荣，赖桃桃，张东站. 一种改进的核增量式更新算法[J]. 计算机工程与应用，2010，46(20)：96-98

(上接第 181 页)

[4] Li H X, Zhou X Z. Risk decision making based on decision-theoretic rough set: a three-way view decision model [J]. International Journal of Computational Intelligence Systems, 2011, 4: 1-11

[5] Liu D, Li T R, Ruan D. Probabilistic model criteria with decision-theoretic rough sets [J]. Information Sciences, 2011, 181 (17): 3709-3722

[6] Liu D, Li T R, Hu P, et al. Multiple-category classification with decision-theoretic rough sets [J]. LNAI, 6401, 2010: 703-710

[7] Liu D, Li H X, Zhou X Z. Two decades' research on decision-theoretic rough sets [C]// Proceedings of 9th IEEE International Conference on Cognitive Informatics. 2010: 968-973

[8] Liu D, Yao Y Y, Li T R. Three-way investment decisions with decision-theoretic rough sets [J]. International Journal of Computational Intelligence Systems, 2011, 4: 66-74

[9] Macmillan F. Risk, Uncertainty and Investment Decision-making in the Upstream Oil and Gas Industry [D]. UK: University of Aberdeen, 2000

[10] Pawlak Z. Rough sets [J]. International Journal of Computer and Information Sciences, 1982, 11: 341-356

[11] Pawlak Z, Wong S K M, Ziarko W. Rough sets: probabilistic versus deterministic approach [J]. International Journal of Man-Machine Studies, 1988, 29: 81-95

[12] Pawlak Z, Skowron A. Rough membership functions [M]. Advances in the Dempster-Shafer Theory of Evidence. New York: John Wiley and Sons, 1994: 251-271

[13] Skowron A, Stepaniuk J. Tolerance approximation spaces [J]. Fundamenta Informaticae, 1996, 27: 245-253

[14] Slezak D. Rough sets and bayes factor [J]. LNCS Transactions on Rough Sets III, 2005: 202-229

[15] Slezak D, Ziarko W. The investigation of the bayesian rough set model [J]. International Journal of Approximate Reasoning, 2005, 40: 81-91

[16] Wong S K M, Ziarko W. Comparison of the probabilistic approximate classification and the fuzzy set model [J]. Fuzzy Sets and Systems, 1987, 21: 357-362

[17] Xie G, Yue W, Wang S Y, et al. Dynamic risk management in petroleum project investment based on a variable precision rough set model [J]. Technological Forecasting & Social Change, 2010, 77: 891-901

[18] Yao Y Y, Zhou B. Naive Bayesian Rough Sets [J]. LNAI, 2010, 6401: 719-726

[19] Yao Y Y. Three-way decision: an interpretation of rules in rough set theory [J]. LNAI, 2009, 5589: 642-649

[20] Yao Y Y. Three-way decisions with probabilistic rough sets [J]. Information Sciences, 2010, 180: 341-353

[21] Yao Y Y. The superiority of three-way decision in probabilistic rough set models [J]. Information Sciences, 2011, 181: 1080-1096

[22] Yao Y Y, Wong S K M. A decision theoretic framework for approximating concepts [J]. International Journal of Man-machine Studies, 1992, 37: 793-809

[23] Yao Y Y. Decision-theoretic rough set models [J]. LNAI, 2007, 4481: 1-12

[24] Yusgiantoro P, Hsiao F S T. Production-sharing contracts and decision making in oil production [J]. Energy Economics, 1993, 10: 245-256

[25] Ziarko W. Variable precision rough set model [J]. Journal of Computer and System Sciences, 1993, 46: 39-59

[26] 李华雄，刘盾，周献中. 决策粗糙集模型研究综述[J]. 重庆邮电大学学报：自然科学版，2010，22(5)：624-630

[27] 李华雄，周献中，李天瑞，等. 决策粗糙集理论及其研究进展[M]. 北京：科学出版社，2011

[28] 刘盾，姚一豫，李天瑞. 三枝决策粗糙集[J]. 计算机科学，2011，38(1)：246-250

[29] 张楠，苗夺谦，岳晓冬. 区间值信息系统的知识约简[J]. 计算机研究与发展，2010，47(8)：1362-1371