形式背景同构判定的等价类算法*)

沈夏炯[1,2] 贾培艳[1] 刘宗田[2]
(河南大学计算机与信息工程学院 河南 开封 475001)[1]
(上海大学计算机工程与科学学院 上海 200072)[2]

摘 要 同构生成概念格是获取概念格的另一途径,而形式背景同构判定是这一方法的前提,也是决定整个算法时间复杂度的关键。本文提出的基于等价类法的形式背景同构判定算法,有效地提高了同构判定的效率。结合形式背景的分解和约简等手段,为概念格的构造提供了一种有实用价值的方法。本文对该方法的原理和算法设计进行了较详细的讨论,并通过实验,验证了算法的正确性和有效性。

关键词 形式概念分析,形式背景,同构,概念格

Context Isomorphism Detection Algorithm: Equivalence Classes Algorithm

SHEN Xia-Jiong[1,2] JIA Pei-Yan[1] LIU Zong-Tian[2]
(School of Computer and Information Engineering, Henan University, Kaifeng 475001)[1]
(School of Computer Engineering and Science, Shanghai University, Shanghai 200072)[2]

Abstract Isomorphic generating is a new way to obtain concept lattices. Isomorphism detecting of context is not only the presupposition of the methodology but also the key procedure to decline the complexity of time. This paper suggests an isomorphism detecting algorithm based on equivalent class, which improves the efficiency of context isomorphism detection. The method provides a practical way to construct concept lattices by combining means such as decomposition and reduction of contexts. The authors discuss the detail of the principles and designs of the algorithm, and verify the validities by implementing the algorithm.

Keywords Formal concept analysis, Context, Iomorphism, Concept lattice

1 引言

形式概念分析[1](FCA,Formal Concept Analysis)被认为是进行知识描述和数据分析的有力工具,而构造概念格是FCA理论中的一个重要问题。目前,已经提出了很多构造算法,包括批处理算法和渐进式算法两大类。批处理算法的代表有Bordat、Chein、Ganter和Nourine等算法;渐进式算法的代表有Godin[2]、Capineto和T. B. Ho等算法。另外还有一些改进算法,例如基于索引树的概念格渐进式构造算法[1,2],扩展概念格的渐进式构造算法[4],改进了的Bordat算法挖掘关联规则[5]。然而,这些算法均是对于一个给定的形式背景从头构造概念格,它们没有研究形式背景之间的关系,也没有研究如何利用这些关系快速地生成概念格。

获取概念格的另一种方法是同构生成,基本原理是同构的形式背景其概念格也是同构的,即根据同构的形式背景的概念格可以生成新的概念格。该方法快捷,但是要求系统中必须存在同构的形式背景,因而形式背景的同构判定成为这一方法的核心问题,而对于该问题,几乎没有文献对此做过研究。

形式背景同构判定尽管源于概念格的同构生成,它却与图同构判定属于同一类问题。图同构判定问题是:给定两个图$G_1=(V_1,E_1)$和$G_2=(V_2,E_2)$,是否存在一个从V_1到V_2的双射函数f,使得$(i,j)\in E_1$当且仅当$(f(i),f(j))\in E_2$。函数f称为G_1到G_2的同构。目前,图同构算法有许多,例如利用关联矩阵计算比较关联度序列的关联度序列法[6]、标识矩阵满足循环1特性(circular 1s property)和连续1特性(consecutive 1s property)的图的并行同构判定[7]、根据无向图的邻接矩阵的特征值来判别图同构关系的无向图判别同构的方法[8]、神经元网络方法[9]等等。由于寻求图的一组不变量(如回路数、树数等)作为判定条件以及通过计算图的邻接矩阵的特征多项式和特征值等进行同构判定的多种思路的试探均未成功,因此多数学者认为图的同构判定问题属于NP-完全问题[10],即它是介于P问题和NP问题之间的一类问题。这类问题,至今尚未找到多项式时间复杂性算法。该问题当前最好时间复杂性是$e^{\sqrt{n\log n}}$[11],然而不能证明这是最优的。对许多被限制的图如平面图,可找到多项式时间复杂度的判定算法[12]。文[13]给出一个方法,把标准的图同构问题转换成一种称为k-PNG(k-Permutation Network Graph)的特殊图的同构问题,并且给出了$O(kn\log kn)$复杂度的算法。

形式背景同构判定问题与采用邻接矩阵的无向图的同构判定问题非常相似,却不完全相同。区别之一是形式背景可看作任意阶数的二值矩阵,而无向图的邻接矩阵是一个二值方阵。所以在某种程度上形式背景同构判定的问题更具有一

*)受国家自然科学基金(60275022)、上海市高等学校青年发展基金(03AQ99)和河南省自然科学基金(0311011700)资助。**沈夏炯** 博士研究生,副教授,研究方向是软件工程、知识发现、分布式/并行计算及分布式存储;**贾培艳** 硕士研究生,研究方向是软件工程、知识发现和分布式/并行计算;**刘宗田** 博士生导师,教授,主要研究领域为人工智能和软件工程。

般性。因此，研究形式背景同构判定算法不仅对概念格的生成具有实际意义，而且对于图同构判定问题也具有理论意义。

本文提出一种基于等价类的变换算法，基本思路是首先把形式背景的对象集进行等价类划分，然后变换时只在同一个等价类中交换对象；对于形式背景的属性集也是如此，从而极大地提高了形式背景同构判定的效率。本文内容分为5节：第2节简要介绍有关形式背景及其同构的基本概念；第3节提出了形式背景同构判定算法—等价类算法，定义了一些相关概念，证明了相关性质，并介绍了算法实现；第4节分析等价类算法；最后是结论性总结。

2 形式背景以及同构的基本概念

为便于下面的讨论，先引入形式背景和同构的概念。

定义1[1] 形式背景被定义为一个三元组 $K=(G, M, I)$，其中 G 和 M 分别是对象集合和属性集合，而 I 是 G 和 M 之间的二元关系，即 $I\subseteq G\times M$，为了表示一个对象 g 和一个属性 m 在关系 I 中，可以写成 gIm 或 $(g, m)\in I$，并且读成“对象 g 具有属性 m”。形式背景可被表示为以对象为行、以属性为列的二值矩阵，其每一元素 a_{jk}（j 为行序，k 为列序，行、列起始序号均为0）可定义为：

$$a_{jk}=\begin{cases}1, & \text{如果对象 } j \text{ 拥有属性 } k\\ 0, & \text{否则}\end{cases}$$

定义2 对于任意一个对象 $g\in G$，定义 $g'=\{m\in M \mid gIm\}$ 表示该对象拥有的属性集。相应地，对于任意 $m\in M$，定义 $m'=\{g\in G \mid gIm\}$ 表示拥有该属性的对象集。

	a	b	c	d	e
1	1	0	0	1	0
2	0	1	1	0	0
3	0	1	1	1	1
4	1	1	0	1	0

图1 一个4×5的形式背景

在图1所示的形式背景中，对象集 G 为{1, 2, 3, 4}，属性集 M 为{a, b, c, d, e}，关系 I 为{(1, a), (1, d), (2, b), (2, c), (3, b), (3, c), (3, d), (3, e), (4, a), (4, b), (4, d)}，并且 $3'=\{b, c, d, e\}$，$a'=\{1, 4\}$。

定义3[1] 对于两个形式背景 (G_1, M_1, I_1) 和 (G_2, M_2, I_2)，如果存在双射 $\alpha: G_1\to G_2$，$\beta: M_1\to M_2$ 使得对于任意的 $g\in G_1$，$m\in M_1$ 有 $gI_1m\Leftrightarrow\alpha(g)I_2\beta(m)$ 成立，那么就称这两个形式背景同构。

为判断两个形式背景是否同构，需要确定能否找到两者之间的上述映射对 (α,β)。一个直观的方法（以下称为直观算法）是：给定形式背景 C_1 和 C_2，对 C_1 做行与行、列与列的交换，得到 C'_1，如果能使 $C'_1=C_2$，则形式背景 C_1 和 C_2 是同构的；否则 C_1 和 C_2 不同构。在最坏情况下判定两个形式背景是否同构需要进行行的全排列和列的全排列，交换的总次数将达到 $r!\times c!$（r 为形式背景的行数，c 为形式背景的列数），该算法的复杂性远远大于指数时间复杂性[6]。

3 形式背景同构判定算法：等价类法

3.1 等价类法的基本思想

为有效降低直观算法的交换次数，本文提出一个新的形式背景同构判定算法：等价类法。其基本思想为，首先，引入等重（weight）关系，并对于形式背景 C_1 和 C_2 分别进行等重划分，然后，对划分后的形式背景的各部分逐一进行判定。对于任意一个形式背景进行等重划分的过程包括两部分，对象划分和属性划分。对象划分在计算每一对象（行）的重的基础上，按照等重关系将对象集划分为等价类，并使得对象等价类按其重的大小排序。为确定形式背景形态上的唯一性，在变换过程中，始终要保持等价类的有序性。属性划分类似。于是，等重划分后的形式背景被对象等价类和属性等价类分解为多个子形式背景，只须对 C_1 和 C_2 的对应子形式背景运用直观算法进行判定。由于采用等重划分，算法的复杂性将大大降低。以下给出了等价类算法需要用到的一些定义和定理。

定义4 设有形式背景 (G, M, I)，对于任意对象 $g\in G$，定义 $\text{weight}(g)=\|g'\|$，即对象 g 的属性集中元素的个数；同样，对于任意属性 $m\in M$，定义 $\text{weight}(m)=\|m'\|$，即属性 m 的对象集中元素的个数。对于 $\forall g_1, g_2\in A\subseteq G$，若 $\text{wieght}(g_1)=\text{weight}(g_2)$，则定义 $\text{weight}(A)=\text{weight}(g_1)$。

例如在图1中，对于形式背景1，对象1到对象4的weight分别为2、2、4和3，属性 a 到属性 e 的weight分别为2、3、2、3和1。

定义5 给定形式背景 (G, M, I)，定义：

$\text{WeightSequence}(G)=\langle\text{weight}(g_1), \text{weight}(g_2), \cdots, \text{weight}(g_i), \cdots \text{weight}(g_k)\rangle$，其中 $g_1, g_2\cdots, g_k\in G$，且满足 $\text{weight}(g_i)>\text{weight}(g_{i+1})$，$i=1, 2, \cdots, k$，$k=\|G\|$。$\text{WeightSequence}(G)$ 表示形式背景全部对象的weight所组成的降序序列。同样定义：

$\text{WeightSequence}(M)=\langle\text{weight}(m_1), \text{weight}(m_2), \cdots, \text{weight}(m_i), \cdots \text{weight}(m_l)\rangle$，其中 $m_1, m_2\cdots, m_l\in M$，且满足 $\text{weight}(m_i)>\text{weight}(m_{i+1})$，$i=1, 2, \cdots, l$，$l=\|M\|$。$\text{WeightSequence}(M)$ 表示形式背景全部属性的weight所组成的降序序列。

对于图1中的形式背景，$\text{WeightSequence}(G)=\langle4,3,2,2\rangle$，$\text{WeightSequence}(M)=\langle3,3,2,2,1\rangle$。

定义6 给定形式背景 (G, M, I)，对于对象及属性来说，可以根据weight定义如下两个等价关系：

$R_G\subseteq G\times G$：{$(g_1, g_2) \mid g_1, g_2\in G$ 且 $\text{Weight}(g_1)=\text{Weight}(g_2)$} 和 $R_M\subseteq M\times M$：{$(m_1, m_2) \mid m_1, m_2\in M$ 且 $\text{Weight}(m_1)=\text{Weight}(m_2)$}。

行等价类集合表示为 G/R_G，列等价类集合表示为 M/R_M。

在图1中，根据等价关系，对象集{1, 2, 3, 4}被划分为3个等价类 $[1]_{R_G}$、$[3]_{R_G}$ 和 $[4]_{R_G}$，而属性集被划分为3个等价类 $[a]_{R_M}$、$[b]_{R_M}$ 和 $[e]_{R_M}$。另外，只有对于同一属性集的对象行讨论行等价类、对于同一对象集的属性列讨论列等价类才有意义。

定义7 给定形式背景 (G, M, I)，对于 $\forall g\in G$，定义

$\text{Value}(g)=\text{BinToDec}(a_1a_2\cdots a_l)$，

称为对象 g 的value，$l=\|M\|$，属性值

$$a_i=\begin{cases}1, & \text{若}(g, m_i)\in I\\ 0, & \text{否则}(i=1, 2, \cdots, l)\end{cases}$$

各属性值的顺序为形式背景中的当前顺序；同样，对于 $\forall m\in M$，定义

Value(m)= BinToDec($b_1b_2\cdots b_k$)，

称为属性 m 的 value，$k=\|G\|$，对象值

$$b_i=\begin{cases}1, & 若(g_i,m)\in I\\ 0, & 否则\end{cases}\qquad(i=1,2,\cdots,k)$$

各对象值的顺序为形式背景中的当前顺序，BinToDec 函数将二进制字符串转化为对应的十进制值。如在图 1 中，value(1)= 18，value(b)= 7。

交换行会影响各列的 value，交换列会影响各行的 value。

定理 1 如果形式背景 $C_1=(G_1, M_1, I_1)$和 $C_2=(G_2, M_2, I_2)$是同构的，则

WeightSequence(G_1)= WeightSequence(G_2)，且

WeightSequence(M_1)= WeightSequence(M_2)。

证明：如果形式背景 C_1 和 C_2 同构，那么我们总能通过行列交换，使两者的对应对象、对应属性处在相同的行、列位置上，也就是说可以得到完全相同的形式背景 C'_1 和 C'_2，很显然，WeightSequence(G_1)= WeightSequence(G_2)，WeightSequence(M_1)= WeightSequence(M_2)是必然满足的，且 C'_1 和 C'_2 各对应属性的 value 值也是相等的。证毕。

定义 8 给定形式背景(G, M, I)，对于 $A\subseteq G$，定义：

ValueSequence(A)=〈Value(g_1)，Value(g_2)，…，Value(g_i)，…，Value(g_s)〉，其中 $g_1, g_2\cdots, g_s\in A$，且满足 Value(g_i)>Value(g_{i+1})，$i=1, 2, \cdots, s$，$s=\|A\|$。ValueSequence(A)表示形式背景对象的某个子集中所有对象的 value 的降序序列。

同样，对于 $B\subseteq M$，

ValueSequence(B)=〈Value(m_1)，Value(m_2)，…，Value(m_i)，…，Value(m_t)〉，其中 $m_1, m_2\cdots, m_t\in B$，且满足 Value(m_i)>Value(m_{i+1})，$i=1, 2, \cdots, t$。ValueSequence(B)则表示形式背景属性的某个子集中所有属性的 value 的降序序列。

另外，虽然交换 A 中某两个对象所对应行，会改变 ValueSequence(B)，交换 B 中某两个属性所对应的列，会改变 ValueSequence(A)，但等价类法只考虑列的交换，且行等价类始终保持有序，所以不必考虑排序时行的交换对列的影响。

定理 2 设有形式背景 $C_1=(G_1, M_1, I_1)$和 $C_2=(G_2, M_2, I_2)$，并满足 WeightSequence(G_1)= WeightSequence(G_2)且 WeightSequence(M_1)= WeightSequence(M_2)，R_{G_1} 是 G_1 上的等重关系，R_{G_2} 是 G_2 上的等重关系。如果对 $\forall E_1\in G_1/R_{G_1}$，都 $\exists E_2\in/G_2/R_{G_2}$，使得：

ValueSequence(E_1)= ValueSequence(E_2)

成立，那么 C_1 和 C_2 是同构的。

证明：对于形式背景 C_1 和 C_2，如果 WeightSequence(G_1)= WeightSequence(G_2)，即这两个序列对应元素相等，明显，对 $\forall E_1\in G_1/R_{G_1}$，必 $\exists E_2\in C_2/R_{G_2}$，满足 weight($E_1$)= weight($E_2$)且 $\|E_1\|=\|E_2\|$；如果 ValueSequence(E_1)= ValueSequence(E_2)，那么根据 value 的定义，E_2 与 E_1 中 value 相同的行必然是相等的，于是 E_1 与 E_2 是相等的；因此，C_1 和 C_2 中 ValueSequence 相等的等价类都是相等的。分别对 C_1 和 C_2 进行行列交换，使其各对象等价类按 weight 降序排列、各属性等价类也按 weight 降序排列，得到 C'_1 和 C'_2，则必然有 $C'_1=C'_2$，故 C_1 和 C_2 是同构的。证毕。

对偶地，R_{M1} 是 M_1 上的等重关系，R_{M2} 是 M_2 上的等重关系，如果对 $\forall F_1\in M_1/R_{R_{M1}}$，都 $\exists F_2\in M_2/R_{M2}$，使得 ValueSequence($F_1$)= ValueSequence($F_2$)，那么 C_1 和 C_2 也是同构的。

3.2 基于等价类的形式背景同构判定算法的设计与实现

定理 1 给出了同构形式背景的一个重要必要条件，定理 2 则给出了形式背景同构判定的充分条件。根据这两个定理，设计等价类算法如下。为形象起见，描述中把对象等价类称为行等价类，把属性等价类称为列等价类。

(1) 对于形式背景 $C_1=(G_1, M_1, I_1)$和 $C_2=(G_2, M_2, I_2)$，如果 WeightSequence(G_1)≠ WeightSequence(G_2)或者 WeightSequence(M_1)≠ WeightSequence(M_2)，那么两者不同构。

(2) 对 C_1、C_2 的行及列进行等重划分，然后根据形式背景 C_1、C_2 行、列等价类的 weight，并分别按降序排列。这一步称为 weight 排序。

对经过 weight 排序的 C_1 和 C_2 中各个行等价类，分别计算类中各行的 value，按 value 值降序排列，这一步称为 value 类内排序，处理后的形式背景记为 C'_1 和 C'_2。

此时检查对 $\forall E_1\in G_1/R_{G_1}$，是否 $\exists E_2\in G_2/R_{G_2}$，使得 ValueSequence($E_1$)= ValueSequence($E_2$)。若是，那么形式背景 C_1、C_2 是同构的，中止算法；否则，仅对 C'_2 继续做如(3)的处理。

(3) 对 C'_2 中某一个列等价类，交换任意两列，每做一次变换，便对各个行等价类按 value 做类内排序，然后计算一次各个行等价类的 ValueSequence，判断是否满足定理 2，如果满足，则形式背景 C_1、C_2 同构，否则继续做列变换，直到对 C'_2 中每一个列等价类都作上述计算。如果最终仍不满足定理 2，那么形式背景 C_1、C_2 不同构。该步处理称为等价类变换。

在整个判定过程中，共提到了三种处理：weight 排序、value 类内排序和等价类变换。前两种比较容易实现。而第三种处理是本算法的核心，使用了递归的思想，以下为等价类变换及比较的过程算法：

```
//对形式背景 context 的某等价类中从 begin 列到 end 列的范围进行变换比较
function AdvancedCompare(context, begin, end)
{
    // bCompare 表示两个形式背景同构判定
    // 的结果，true 表示同构，false 表示不同(构)
    bool bCompare = false;
    将 context 复制到 backup;
    for (i = begin; i ≤ end; i++)
    {
        if (i ≠ begin)
        {
            将 backup 复制到 context;
            交换 context 的第 i 列和第 begin 列;
        }
        // 如果 begin 不是该等价类的最后一 列则递归
        if (end > begin)
        {
            bCompare=
             AdvancedCompare(context, begin+1, end);
            if (bCompare)break;
        }
        else
        {
            // Exitequivalent 函数是找该等价类的 end 列后的下一个等价类(至少两列)，如果有则返回该等价类中最靠近 end 的列，否则返回- end
            nextbegin = Exitequivalent (end);
            if (nextbegin > 0)
            {
                // End 函数求该等价类的下 一个等价类的末列序号
            nextend = End (nextbegin);
            bCompare=
            AdvancedCompare (context, nextbegin, nextend);
            if (bCompare)break;
        }
```

```
        // 否则,比较 aValue,bvalue(分别存放比较形式背景、被比
        较形式背景各行 value 的一维数组)
        else
        {
            按 value 对 context 的等价类中各行排序;
            获得重新排序后 context 的 avalue ;
            // compare 函数比较
            aValue,// bvalue
            bCompare=compare
            (aValue,bvalue);
            // 如果两者对应值相同,同// 构,返回结果,跳出循环
            if (bCompare)break;
        }
    }
  }
  // 返回结果值,当然如果此时返回,则表明 a 未得到与被比较形式
  背景相同的形式,返回值为 false
  return bCompare;
}
```

4 算法实验与分析

假设 $r\times c$ 的形式背景中,r 行中存在 x 个等价类,第 i 个等价类中有 m_i 行($0<i<x$, $0<m_i<r$),$m_1+m_2+\cdots+m_{x-1}+m_x=r$,$c$ 列中存在 y 个等价类,第 j 个等价类中有 n_j 个元素($0<j\leqslant y$, $0<n_j\leqslant c$),$n_1+n_2+\cdots+n_{y-1}+n_y=c$,基于等价类的判定算法仅对同一等价类中的列作变换,而等价类中的行是根据列变换后 value 的变换而重新在等价类中重新确定行序。那么,列最多做 $w=n_1!\times n_2!\cdots\times n_y!$ 次变换,很明显 $w<=c!$,只有当 $y=1$ 的最坏情况下,也就是形式背景所有列属于同一个等价类的时候,用该算法处理时,列才会交换 $n!$ 次;否则,等价类个数 y 越大,$\max(n_1, n_2, \cdots, n_y)$越小,$w$ 越小。每一次列变换后,需要根据各行的 value 的可能变化而重新将各等价类内的行进行排序,由于行排序是采用冒泡排序,那么,行最多做 $v=m_1(m_1-1)/2+m_2(m_2-1)/2+\cdots+m_x(m_x-1)/2$ 次变换,$v\ll r!$,当 $x=1$ 的最坏情况下,也就是形式背景所有行属于同一个等价类的时候,用该算法处理时,行交换 $m(m-1)/2$ 次,否则,等价类个数 x 越大,$\max(n_1, n_2, \cdots, n_y)$越小,$v$ 也越小。

令 $m_0=\mathrm{Max}(m_1, m_2, \cdots, m_x)$,$n_0=\mathrm{Max}(n_1, n_2, \cdots, n_y)$,可以得出该算法的时间复杂度为 $O(m_0^2\times n_0!)$,而直接算法的时间复杂度为 $O(m!\times n!)$,显然是优于直接算法的,并且随着 x、y 的增大以及 m_0 和 n_0 的减小,其时间复杂度急剧减少。也就是说该算法更适合于列中存在多个等价类的情况。即使两个形式背景不同构,最多需要做$(m_1(m_1-1)/2+m_2(m_2-1)/2+\cdots+m_x(m_x-1)/2)\times(n_1!\times n_2!\times\cdots\times n_y!)$交换,而不是 $m!\times n!$ 次交换。

以下实验分别用直观算法和等价类算法对 7 大组随机生成的形式背景进行判定。考虑到直观算法时间复杂度太高,这 7 大组形式背景的规模(属性数)分别取 1 至 7;每大组包括 5 小组形式背景,每小组分别对 6 对形式背景进行判定,以这 6 个结果的算术平均值作为小组的结果,然后以 5 个小组结果的算术平均值作为大组的结果。图 2 中的每个点代表一个大组的结果,纵坐标表示列变换次数;横坐标表示形式背景规模。

由于两种算法性能差异过于悬殊,为了能得到视觉上有所区别的图,我们对直观算法实验数据的局部略作变形处理,即对 4×4 形式背景的列变换次数除以 10,对 5×5 的列变换次数除以 100,对 6×6 的除以 1000,对 7×7 的除以 10000;而等价类算法的数据不变。图中直观算法的曲线为变形处理之后的结果,等价类算法为实际数据。表 1 列出了全部真实的实验数据和经过变形的数据。

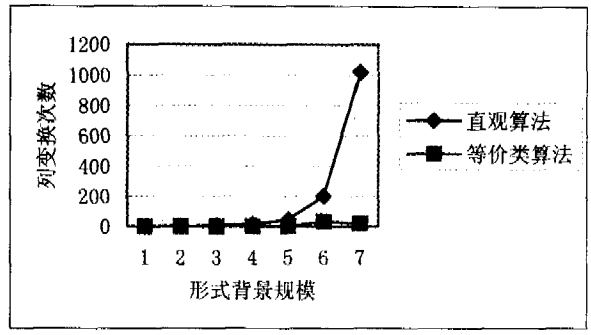

图 2 两种算法比较次数的对比

表 1 两种算法实验数据和部分变形处理结果

		0×0	1×1	2×2	3×3	4×4	5×5	6×6	7×7
直观算法	实际	0	1	2	6	140	4580	201693	10208295
	变形	0	1	2	6	14	45.8	201.693	1020.8295
等价类法		0	1	1	1	2	3	35	22

从图中可以看出,随着形式背景规模的增大,直观算法的列变换次数急剧增加,与等价类算法性能的差距是非常明显的。在对 12×12 形式背景做同构判定时,直观算法在 8 小时内做了 2 亿多次列变换仍未得出结果,而等价类算法仅用不到一秒变换了 44 次就得出结果。虽然这两个形式背景具有一定的特殊性,但实验结果表明:即使出现最坏情况,即等价类算法进行判定而列的变换达到 $n!$ 的时候,等价类算法使用的时间也远远少于直观算法,这是因为所有的行始终保持有序,使得行变换的总次数远远小于 $r!$。

结论 等价类算法的独特之处就在于它引入了 weight、value 等概念,并按照 weight 相等的关系对形式背景的对象集和属性集分别进行等价类划分,缩小了行、列变换的范围,有效地减少了行、列变换的次数,同时也减少了比较的次数。并且在进行行、列变换之前,对形式背景按 weight 排序,在进行比较之前,对形式背景又按 value 对等价类中各元素排序,这样便不需再对行做变换处理。通过与直观算法的比较,证明了该算法在时间复杂性和空间复杂性上都具有的优良性能。该算法还有可以改进的地方,例如改进排序算法、采用非递归算法以及进一步减小行列交换范围等等,都可进一步提高算法的效率。

最后指出,等价类判定法适用于采用邻接矩阵的无向图同构判定问题,由于无向图的邻接矩阵是对称方阵,因此还有进一步简化算法的余地。另外,本算法适合并行计算。因为在做列变换时,行中对应部分的交换以及比较是可以独立进行的,如果使用并行计算机系统来实现本算法,计算速度将有更大的提高。

参 考 文 献

1 Ganter B, Wille R. Formal Concept Analysis: Mathematical Foundations [M]. Berlin: Springer Verlag, 1999

2 Godin R, Missaoui R, Alaoui H. Incremental concept formation algorithms based on Galois(concept) lattices. Computational Intelligence, 1995, 11(2):246~267

(下转第 155 页)

后三类合称为不良贷款。本文的应用背景是建设银行大连市分行个人住房贷款风险分析，并建立了个人房贷风险分析模型。

涉及个人贷款的特征数据很多，我们从相关的个贷系统的个人基本信息表、个人贷款申请表和信贷管理信息系统的个贷合同信息表中抽取出了相关的特征数据，一共32个。通过仔细分析，并通过属性约简技术，我们过滤了一些对个贷没有影响或影响很小的一些特征属性(数据)，得到最终用于分析的特征属性(数据)15个。

原始实验数据共3423条，实验前首先进行了数据预处理。从中随机抽取2/3样本(2282个)作为训练数据，剩余的1/3样本(1141个)作为测试数据。重复抽取10次，最后得到10组数据。我们利用这10组数据针对1-a-a算法和FC-SVM算法，分别采用径向基核、多项式核、线性核进行了训练和预测。实验结果如表2所示，表中数据为分类正确率(%)。

表2 个人房贷风险分析实验结果

实验组别	RBF核 ($\sigma^2=18, C=2$)		多项式核 ($d=3, C=0.05$)		线性核 ($C=0.05$)	
	1-a-a	FC-SVM	1-a-a	FC-SVM	1-a-a	FC-SVM
1	0.85	0.93	0.80	0.87	0.74	0.83
2	0.90	0.94	0.82	0.90	0.76	0.84
3	0.84	0.91	0.79	0.83	0.72	0.81
4	0.89	0.90	0.82	0.86	0.75	0.82
5	0.85	0.92	0.78	0.82	0.74	0.83
6	0.80	0.91	0.76	0.86	0.72	0.82
7	0.82	0.93	0.77	0.83	0.76	0.83
8	0.85	0.94	0.79	0.83	0.71	0.79
9	0.91	0.91	0.81	0.85	0.7	0.82
10	0.86	0.93	0.83	0.86	0.75	0.82
平均	0.857	0.922	0.797	0.85	0.74	0.822

在信用风险评估中，我们并不能判定某个客户是绝对好客户，该客户一定能按时偿还所有贷款；我们也不能说另一个就是绝对的差客户，他一定会拖欠债务。客户信用的评价是通过概率值来确定的，他们只是在违约概率上的大小不同，即使一个最佳信用客户也完全可能拖欠贷款。从试验结果中分析，我们发现FC-SVM方法具有较高的正确分类率，而且在支持向量机中使用径向基核的准确度比多项式核和线性核要高。

结束语 本文提出了一种新的基于直接构造多类SVM分类器的模糊补偿多类SVM算法：FC-SVM算法，重构了优化问题及其约束条件，重构了Lagrange公式，并进行了推导。我们将该算法成功地应用到了中国建设银行大连分行个人住房贷款风险分析系统，并建立了个人房贷风险分析模型，通过试验数据分析比较，算法取得了比较满意的效果。但我们的算法在进一步优化时也存在一个问题，即参数太多。这是我们下一步的工作。

参考文献

1 Vapnik V N. Statistical Learning Theory. New York: John Wiley & Sons, 1998

2 Hsu C W, Lin C J. A comparison of methods for multi-class support vector machines. IEEE Transactions on Neural Networks, 2002, 13(2): 415～425

3 Burges J C. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 1998, 2(2): 121～167

4 XU Jian-hua, ZHANG Xue-gong, LI Yan-da. Advances in Support Vector Machines. Control and Decision, 2004, 19(5):481～484

5 Weston J, Watkins C. Multi-class support vector machines: [Technical Report SD2TR298204]. Department of Computer Science, Royal Holloway University of London ,1998

6 Platt J C, Cristianini N, Shawe-Taylor J. Large margin DAG's for multiclass classification. Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press, 2000, 12: 547～553

7 Platt J. Fast t raining of support vector machines using sequential minimal optimization [A]. Advances in Kernel Methods -Support Vector L earning [C]. Cambridge: M IT Press, 1999. 185～208

8 Vapnik V N. The Nature of Statistical Learning Theory [M]. 2nd edition. New York: Sp ringer-Verlag, 1999

9 Suykens J, Vandewalle J. Least squares support vector machines [J]. Neural Processing Letters, 1999, 9(3): 293～300

10 Scholkopf B, Smola A J, Williamson R C, et al. New support vector algorithms [J]. Neural Computation, 2000, 12(5): 1207～1245

11 Mangasarian O L. Generalized support vector machines [A]. Advances in Large Margin Classifiers [C]. Cambridge: MIT Press, 2000. 135～146

12 Inoue T, Abe S. Fuzzy Support Vector Machines for Pattern Classification. In: Proceedings of International Joint Conference on Neural Networks, 2001. 1449～1454

13 Lin C F, Wang S D. Fuzzy Support Vector Machines. IEEE Transactions on Neural Networks, 2002, 13(2): 464～471

14 Sun Z H, Sun Y X. Fuzzy Support Vector Machine for Regression Estimation. IEEE International Conference on Systems, Man and Cybernetics, 2003, 4: 3336～3341

15 Hong D H, Hwang C. Support Vector Fuzzy Regression Machines. Fuzzy Set s and Systems, 2003, 138(2): 271～281

16 Huang Z,Chen H C, Hsu C J, et al. Credit rating analysis with support vector machines and neural networks: a market comparative study. Decision Support Systems,2004,37:543～558

17 http://www.ics.uci.edu/～mlearn/MLRepository.html

18 http://www.cs.wisc.edu/musicant/data/ndc,1998

(上接第151页)

3 谢志鹏，刘宗田．概念格的快速渐进式构造算法．计算机学报，2002(5)：490～496

4 简宋全，胡学钢，蒋美华．扩展概念格的渐进式构造．计算机工程与应用，2001，15：132～134

5 胡可云，陆玉昌，石纯一．基于概念格的分类和关联规则的集成挖掘方法．软件学报，2000，11(11)：1478～1484

6 李锋，李晓燕．图的同构判定算法：关联度序列法及其应用．复旦学报(自然科学版)，2001，40(3)

7 Chen Lin. Parallel and Distributed Systems. IEEE Transactions on 1996,7(3):308～319

8 陈峻，殷新春．无向图同构判定的并行算法．计算机工程，2002，6(28)

9 Funabiki N,Kitamichi J. Three-stage Greedy and Neural-network Approach for Subgraph Isomorphism Problem. In: Proceedings of the 1998 IEEE International Conference on System, Man and Cybernetics, IEEE, Piscataway, NJ, USA, 1998, 2:1892～1897

10 Karp R M. Redacibility among combinatorial problems [M]. New York :Plenum Press, 1972. 83～85

11 Babai L, Luks E. Canonical labeling of graphs. In: Proc. 15th ACM Symp. on Theory of Computing, 1983. 171～183

12 Hopcroft J E, Tarjan R E. A V^2 algorithm for determining isomorphism of planar graphs. Information Processing Letters, 1971. 32～34

13 Sfudhar A. A Fast Algorithm for Testing Isomorphism of Permutation Networks. Computers, IEEE Transactions on, 1989, 38 (6):903～909