前向 Beam 搜索粗糙集属性约简算法*)

杨 胜[1] 施鹏飞[2]

(湖南大学计算机与通信学院 长沙 410082)[1] (上海交通大学图像处理与模式识别研究所 上海 200030)[2]

摘 要 从属性集互信息的角度分析了粗糙集理论的属性约简问题。粗糙集属性约简通常采用 Best-first 启发式搜索。本文运用属性集互信息作为属性约简度量,提出了前向 Beam 搜索粗糙集属性约简算法。实验表明,属性约简算法具有良好的运行效果。

关键词 粗糙集,属性约简,互信息,Beam 搜索

A Forward-beam Search-based Attribute Reduction Algorithm for Rough Set

YANG Sheng[1] SHI Peng-Fei[2]

(School of Computer and Communication of Hunan University, Changsha 410082)[1]

(Institute of Image Processing and Pattern Recognition of Shanghai Jiaotong University, Shanghai 200030)[2]

Abstract The attribute reduction problem for rough set is analyzed by the mutual information of attribute set. Attribute reduction of rough set often employs a Best-first search. Based on mutual information of attribute set, a forward Beam search-based attribute reduction algorithm for rough set is presented. Experiments show that the new algorithm yields satisfying attribute reduction results.

Keywords Rough set, Attribute reduction, Mutual information, Beam search

1 引言

Z. Pawlak 提出了粗糙集理论[1],它通过属性约简和值约简来得到分类规则。粗糙集理论作为一种处理不精确性信息的分类问题工具,已被广泛应用于数据挖掘、机器学习、人工智能以及故障诊断等领域。属性约简是粗糙集理论分类规则提取过程中的一个基本操作,它是指在保持初始属性集的分类能力的前提下删除不相关和冗余的属性。在属性约简的基础上进行值约简,得到分类规则。

最小属性约简是一个 NP-hard 问题[2]。得到最小属性约简最直接的方法就是采用完全搜索算法,如分支界限法[3,4]等。然而,随着数据集维数的增大,必将导致算法运行时间的增加。因此,属性约简倾向于采用启发式算法。不过,启发式通常是采用 Best-first 方式,即每一步选取具有最好的评价函数值的属性集作为进一步搜索的出发点,虽然使得搜索范围大大减小,但同时降低了约简质量。

文[5,6]提出了粗糙集理论概念的信息熵表达。本文则从属性集互信息的角度分析粗糙集理论的属性约简问题,提出了基于 Beam 搜索[7]的前向粗糙集属性约简算法。Beam 搜索算法可看作是完全搜索算法的一个简化,但它和 Best-first 搜索一样是一个启发式搜索算法,具有多项式时间复杂度的特点。

2 理论框架

2.1 基于互信息的粗糙集概念分析

定义 1 信息系统 $IS=\langle U, R, V, \xi\rangle$,$U$ 为论域;$R=F\cup D$ 为属性集合,$F=\{f_1,f_2,\cdots,f_p\}$ 为条件属性集,$p=|F|$,$D=\{P\}$ 为决策属性集,P 为决策属性;$V=\bigcup_{f\in R}V_f$,V_f 表示属性 f 的值域;ξ: $U\times R\to V$ 为一个信息函数。

定义 2 $A\subseteq F$,$IND(A)$ 表示 A 上的不可分辨关系。$f\in A$,如果 $IND(A)=IND(A-\{f\})$,则称 f 在 A 中为可去除属性;否则,称 f 在 A 中为不可去除属性(或必要属性)。所有 A 中的必要属性组成的集合称为 A 的核,记为 $CORE(A)$。

A ($A\subseteq F$)可看作为定义在 U 的子集组成的 σ-代数上的一个随机变量,设 A 在 U 上导出的划分为 Y,$U/IND(A)=Y=\{Y_1,Y_2,\cdots,Y_a\}$。其概率分布可表示如下:

$$[Y;p(\cdot)]=\begin{bmatrix} Y_1 & Y_2 & \cdots & Y_a \\ p(Y_1) & p(Y_2) & \cdots & p(Y_a)\end{bmatrix}$$

其中,$p(\cdot)$表示概率密度函数,$p(Y_j)=|Y_j|/|U|$,$j=1,\cdots,a$,$|\cdot|$表示基数。

定义 3 $A\subseteq F$,如果任意属性 $f\in A$,f 在 A 中为必要属性,则称 A 为独立的;否则,称 A 为相依的。

定义 4 $A\subseteq B\subseteq F$,如果 $IND(A)=IND(B)$,称 A 为 B 的一个等价属性子集;如果 $IND(A)=IND(B)$,且 A 为独立的,则称 A 为 B 的一个约简,记作 $RED(B)$。有 $CORE(B)=\cap RED(B)$。

定义 5 条件属性集 F 与决策属性 P 之间的互信息记为:

$$I(F;P)=I(P)-E(P|F) \tag{1}$$

式(1)中,$I(F;P)$为 F 和 P 之间的互信息,$I(P)$为类属性 P 的期望信息(熵),$E(P|F)$为类属性 P 的条件熵。

设 P 和 F 在 U 上导出的划分分别为 $U/IND(P)=\{X_1,X_2,\cdots,X_u\}$($u$ 表示类数),$U/IND(F)=\{Y_1,Y_2,\cdots,Y_v\}$。$I(P)$和 $E(P|F)$可以分别由式(2)和式(3)计算如下:

$$I(P)=-\sum_{i=1}^{u}p(X_i)\log_2 p(X_i)=-\sum_{i=1}^{u}\frac{|X_i|}{|U|}\log_2\frac{|X_i|}{|U|} \tag{2}$$

$$E(P|F)=-\sum_{j=1}^{v}p(Y_j)\sum_{i=1}^{u}p(X_i|Y_j)\log_2(p(X_i|Y_j))=-\sum_{j=1}^{v}\frac{|Y_i|}{|U|}\sum_{i=1}^{u}\frac{|X_i\cap Y_j|}{|Y_j|}\log_2\frac{|X_i\cap Y_j|}{|Y_j|} \tag{3}$$

$$=-\sum_{j=1}^{v}\sum_{i=1}^{u}\frac{|X_i\cap Y_j|}{|U|}\log_2\frac{|X_i\cap Y_j|}{|Y_j|}=-\sum_{i=1}^{u}\sum_{j=1}^{v}\frac{|X_i\cap Y_j|}{|U|}$$

*)国家自然科学基金(No. 60075007)。

$\log_2 \frac{|X_i \cap Y_j|}{|Y_j|}$

单调性 如果 $A \subseteq B \subseteq F$，则 $I(A;P) \leqslant I(B;P)$[8]。

单调性说明任何属性子集的互信息不小于其任何子集的互信息。

引理 1 如果 $A \subseteq B \subseteq F$，则 $IND(A)=IND(B) \Leftrightarrow E(P|A)=E(P|B)$。

证明：充分性（$IND(A)=IND(B) \Rightarrow E(P|A)=E(P|B)$）

$A \subseteq B$ 且 $IND(A)=IND(B)$，说明 A 和 B 导出完全相同的划分。设 $U/IND(P)=\{X_1,X_2,\cdots,X_u\}$，$U/IND(A)=U/IND(B)=\{Y_1,Y_2,\cdots,Y_v\}$，根据式(3)，易证 $E(P|A)=E(P|B)$。

必要性（$IND(A) \neq IND(B) \Rightarrow E(P|A) \neq E(P|B)$ 或者 $E(P|A)=E(P|B) \Rightarrow IND(A)=IND(B)$）。取 $IND(A) \neq IND(B)$，设 $U/IND(P)=\{X_1,X_2,\cdots,X_u\}$，$U/IND(A)=\{YA_1,YA_2,\cdots,YA_{ua}\}$，$U/IND(B)=\{YB_1,YB_2,\cdots,YB_{ub}\}$。因为 $A \subseteq B$，根据属性子集在 U 上划分的定义，B 在 U 上划分的可看作是属性子集 $B\text{-}A$ 对 A 在 U 上划分的基础上的进一步细分，因此至少存在一个等价类被属性子集 $B\text{-}A$ 所划分。设等价类 YA_j（$YA_j \in U/IND(A)$）被划分为 $YB_{j1},YB_{j2},\cdots,YB_{jb}$（$YB_{j1},YB_{j2},\cdots,YB_{jb} \in U/IND(B)$），得

$$YB_{j1} \cup YB_{j2} \cup \cdots \cup YB_{jb} = YA_j \tag{4}$$

$$|YB_{j1}|+|YB_{j2}|+\cdots+|YB_{jb}|=|YA_j| \tag{5}$$

$$(YB_{j1} \cap X_i) \cup (YB_{j2} \cap X_i) \cup \cdots \cup (YB_{jb} \cap X_i) = (YA_j \cap X_i) \tag{6}$$

$$|YB_{j1} \cap X_i|+|YB_{j2} \cap X_i|+\cdots+|YB_{jb} \cap X_i|=|YA_j \cap X_i| \tag{7}$$

$$E(P|A)-E(P|B)=\sum_{j=1}^{v}\sum_{i=1}^{u}\left(\sum_{k=1}^{jb}\frac{|X_i \cap YB_{jk}|}{|U|}\log_2\frac{|X_i \cap YB_{jk}|}{|YB_{jk}|}-\frac{|X_i \cap YA_j|}{|U|}\log_2\frac{|X_i \cap YA_j|}{|YA_j|}\right) \tag{8}$$

又因为

$$\frac{|X_i \cap YB_{j1}|}{|U|}\log_2\frac{|X_i \cap YB_{j1}|}{|YB_{j1}|}+\frac{|X_i \cap YB_{j2}|}{|U|}\log_2\frac{|X_i \cap YB_{j2}|}{|YB_{j2}|}-(|X_i \cap YB_{j1}|+X_i \cap YB_{j1}|)\log_2\frac{|X_i \cap YB_{j1}|+|X_i \cap YB_{j1}|}{|YB_{j1}|+|YB_{j2}|}>0 \tag{9}$$

通过递推，得

$$\sum_{k=1}^{jb}\frac{|X_i \cap YB_{jk}|}{|U|}\log_2\frac{|X_i \cap YB_{jk}|}{|YB_{jk}|}-\frac{|X_i \cap YA_j|}{|U|}\log_2\frac{|X_i \cap YA_j|}{|YA_j|}>0 \tag{10}$$

因此，等式(8)>0，$E(P|A) \neq E(P|B)$，所以必要性成立。

综合可证引理 1 成立。

定理 1 $A \subseteq B \subseteq F$，$IND(A)=IND(B) \Leftrightarrow I(A;P)=I(B;P)$。

证明：由 $IND(A)=IND(B) \Leftrightarrow E(P|A)=E(P|B)$，$IND(A)=IND(B) \Leftrightarrow I(P)-E(P|A)=I(P)-E(P|B)$，有 $IND(A)=IND(B) \Leftrightarrow I(A;P)=I(B;P)$。

推论 1 $A \subseteq B \subseteq F$，$I(A;P)=I(B;P)$，则 A 为 B 的等价属性子集。

定理 2 $A \subseteq B \subseteq F$，如果 $I(A;P)=I(B;P)$，且 A 独立，是 A 为 B 的一个约简的充分必要条件。

根据定义 4 和定理 1 简单可证。

定理 3 $A \subseteq F$，如果 $I(A;P)>I(A-\{f \mid f \in A\};P) \Leftrightarrow f$ 为 A 的必要属性。

根据单调性，定义 2 和定理 1 简单可证。

定理 4 $A \subseteq B \subseteq F$，如果 $I(A;P)=I(B;P)$，则 B 的必要属性也是 A 的必要属性。

证明：设 $f(f \in B)$ 是 B 的一个必要属性，$I(B-\{f\};P)<I(B;P)$，如果 $f \notin A$，则 $A \subseteq B-\{f\}$，根据单调性得 $I(A;P)<I(B;P)$，与条件矛盾，所以 f 必定属于 A；

又因为 $I(B-\{f\};P)<I(B;P)=I(A;P)$，$A-\{f\} \subseteq B-\{f\}$，因此 $I(A-\{f\};P) \leqslant I(B-\{f\};P)<I(A;P)$，所以 f 为 A 的必要属性。

根据以上从互信息的角度对粗糙集概念的分析和讨论，可知属性约简也就是要找到某个尽量小的属性子集 A，使得 $I(A;P)=I(F;P)$。

3 属性约简算法

过去通常采用完全搜索算法或者 Best-first 启发式搜索算法。相对于“树无限宽度搜索”的分支界限算法，Beam 搜索是一个“树有限宽度搜索”算法，其树搜索宽度设为 M，称为 Beam 宽度[7]。图 1 为一个 Beam 搜索示意图，Beam 宽度为 2；在前向增加的属性约简过程中，图中每个节点表示一个属性集，每个节点比它的上一层中的节点多一个属性；每一层中有两个最好的满足优化条件的树节点作为下一步搜索的出发点，来做进一步搜索，直到满足搜索停止条件。而当 $M=1$ 时，它就是一个典型的 Best-first 搜索。

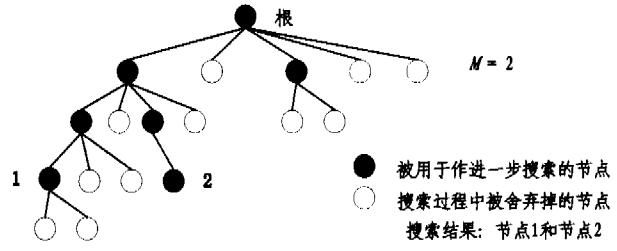

图 1 Beam 搜索示意图

粗糙集属性约简就是要搜索到初始属性集 F 的尽可能小的等价属性子集。根据以上分析，互信息越大，属性子集的划分分类属性的能力越强。选择互信息做启发式，提出了基于 Beam 搜索的前向属性约简算法，其基本思想如下：从 $\{\Phi\}$ 出发，根据定理 3 找到核（核对应图 1 中的根）；在此基础上每次增加一个属性，选取 M 个互信息大的属性子集作为下一步搜索的起点；以此类推，直到找到 F 的等价属性子集为止。这个方法称为基于 Beam 搜索的保持（初始属性集 F）互信息最大化互信息属性约简方法（BEam Maintaining Mutual Information and MAximizing Mutual Information，BEMMIMAMI）。所谓前向是指 BEMMIMAMI 总是不断增加属性，其具体步骤如下：

算法 1 BEMMIMAMI

输入：*IS*-信息系统

F-初始属性集

Beam-Beam 队列

M-Beam 宽度

Queue-一个队列 //长度为 M^2

输出：F 的小的等价属性子集

步骤：Step1 计算 $I(F;P)$；

Step2 根据互信息性质，计算核属性集 *Core*，*Core* 入队列 *Queue*；

Step3 如果 $I(F;P)=I(Core;P)$，转 Step7；否则，Next；

Step4 从 *Queue* 中找到 M 个互信息较大的属性子集，入队列 *Beam*，清空 *Queue*；

Step5 对于 *Beam* 中的每一个属性子集，从其父属性子集中找到 M 个互信息较大的父属性子集，入队列 *Queue*，并且清空 *Beam*；

//父属性子集指增加一个属性

Step6 如果 *Queue* 中有 F 的等价属性子集，Next；否则，转 Step4；

Step7 输出 *Queue* 中所有 F 的等价属性子集。

当 $M=1$ 时，它就是 MIBARK 算法[9]。BEMMIMAMI 属性约简算法的运行时间与两个因素有关系：(1)属性子集互信息的计算；(2) 搜索空间，即被评价的属性子集的个数。一个属性子集评价的时间取决于属性子集对 U 的划分，采用散列法来进行划分。散列函数如式(11)，MOD 表示求 $\sum_{i=1}^{q} i * vf_i$ 除以 L 的余数，vf_i 为属性 f_i 的属性值(整数)，L 为散列表长度，所以属性子集评价的时间复杂度为 $O(m)$，m 表示样本集大小。

$$Hash(A)=\mathrm{MOD}\left(\sum_{i=1}^{q} i * vf_i,\ L\right) \tag{11}$$

BEMMIMAMI 算法被评价的属性子集个数不大于 $p+1+1+0.5*M*p*(p+1)$，式中第一个 p 表示求核时评价的属性子集数，第一个'1'表示求 $I(F;P)$，第二个'1'表示求 $I(Core;P)$，$0.5*M*p*(p+1)$ 表示搜索树生长过程中评价的节点数的上界，因此 BEMMIMAMI 的时间复杂度为 $O(mMp^2)$。BEMMIMAMI 中 M 的取值范围为$[1,\infty]$。当 M 取∞，BEMMIMAMI 就是一个前向完全搜索算法。M 越大，运行时间越长，因此必须选择一个合适的 Beam 宽度来平衡运算时间和属性约简质量。M 的选取原则如下：从小到大选取 M；m 和 p 的值越小，M 的取值可以越大。

4 实验

4.1 实验设计

实验将从 3 个方面对 BEMMIMAMI 算法进行验证：

1) 好的属性约简结果；

2) 较短的运算时间和较小的搜索空间；

3) 运算结果适用于分类规则获取算法。

表 1 测试数据集信息

Dataset	m	p	u
Corral	128	6	2
Monk1	432	6	2
Monk3	432	6	2
Parity5+5	1024	10	2
Parity5+2	1024	10	2
Vote	435	16	2
Lenses	24	4	3
Zoo	101	16	7
Mushroom	8124	22	2
Sonar	208	60	2

m 表示样本集大小，p 表示 $|F|$，u 表示类数。

实验共选用了 10 个 UCI 分类问题数据集，它们是：Corral、Monk1、Monk3、Parity5＋5、Parity5＋2、Vote、Lenses、Zoo、Mushroom、Sonar，数据集信息如表 1 所示。所有连续属性值被离散化。Parity5＋2 中，$f_1=f_6$，$f_2=f_7$，所以必有冗余属性。BEMMIMAMI 算法的 Beam 宽度 M 分别取 $2p$，p 和 1。作为一个比较，FOCUS 和 ABB 算法被选择用于做属性约简。FOCUS 算法是一个前向完全搜索算法，以 F 的互信息作为界。ABB 算法是一个分支界限算法采用完全搜索策略，要求属性约简度量必须具有单调性。因此，可以选择属性集的互信息(具有单调性)作为属性约简度量，而 F 的互信息作为 ABB 的界，这样 ABB 算法可以用于属性约简。属性约简后采用十折交叉校验来考察一个启发式粗糙集值约简方法[10]的分类错误率。

4.2 实验结果及分析

FOCUS 和 ABB 算法的属性约简结果如表 2 所示。ABB 算法可以求出所有 F 的最小等价属性子集，而 FOCUS 在搜索到一个时就停止搜索了，所以只有一个 F 的最小等价属性子集。对于大属性集数据集($p>20$)，ABB 算法由于搜索时间过长而不适用。表 3 为 BEMMIMAMI 算法属性约简结

表 2 FOCUS 和 ABB 算法运算结果

Dataset	ABB		FOCUS	
	AS	r	*AS*	r
Corral	$\{f_1\text{-}f_4\}$	4	$\{f_1\text{-}f_4\}$	4
Monk1	$\{f_1,f_2,f_5\}$	3	$\{f_1,f_2,f_5\}$	3
Monk3	$\{f_2,f_4,f_5\}$	3	$\{f_2,f_4,f_5\}$	3
Parity5+5	$\{f_2\text{-}f_4,f_6,f_8\}$	5	$\{f_2\text{-}f_4,f_6,f_8\}$	5
Parity5+2	$\{f_1\text{-}f_5\}^{(1)}$ $\{f_1,f_3\text{-}f_5,f_7\}^{(2)}$ $\{f_2\text{-}f_6\}^{(3)}$ $\{f_3\text{-}f_7\}^{(4)}$	5	$\{f_1\text{-}f_5\}$	5
Vote	$\{f_1\text{-}f_4,f_9,f_{11},f_{13},f_{15},f_{16}\}$	9	$\{f_1\text{-}f_4,f_9,f_{11},f_{13},f_{15},f_{16}\}$	9
Lenses	$\{f_1\text{-}f_4\}$	4	$\{f_1\text{-}f_4\}$	4
Zoo	$\{f_3,f_4,f_6,f_8,f_{13}\}^{(1)}$ $\{f_3,f_4,f_6,f_9,f_{13}\}^{(2)}$ $\{f_3,f_6,f_8,f_{10},f_{13}\}^{(3)}$ $\{f_4,f_6,f_8,f_{12},f_{13}\}^{(4)}$ $\{f_3,f_6,f_8,f_{13},f_{16}\}^{(5)}$ $\{f_4,f_6,f_9,f_{12},f_{13}\}^{(6)}$ $\{f_3,f_6,f_9,f_{13},f_{16}\}^{(7)}$	5	$\{f_3,f_4,f_6,f_8,f_{13}\}$	5
Mushroom	-	-	$\{f_3,f_4,f_{12},f_{22}\}$	4
Sonar	-	-	$\{f_1,f_6,f_{20},f_{43}\}$	4

AS 为属性约简结果，r 为属性约简结果的大小，"-"表示搜索时间大于 2 小时，认为算法不适合这个数据集的属性约简。

果，可以看出 BEMMIMAMI 算法几乎得到与 ABB 算法相同的属性约简结果。对于 Sonar，BEMMIMAMI 算法分别搜索到了 661($M=p$)和 758($M=2p$)个 F 的较小的等价属性子集。随着 M 值的增大，属性约简结果越来越好。

表 3 BEMMIMAMI 算法属性约简结果

Dataset	BEMMIMAMI ($M=2p$)		BEMMIMAMI ($M=p$)		BEMMIMAMI ($M=1$)	
	AS	*r*	*AS*	*r*	*AS*	*r*
Corral	$\{f_1\text{-}f_4\}$	4	$\{f_1\text{-}f_4\}$	4	$\{f_1\text{-}f_4\}$	4
Monk1	$\{f_1,f_2,f_5\}$	3	$\{f_1,f_2,f_5\}$	3	$\{f_1,f_2,f_5\}$	3
Monk3	$\{f_2,f_4,f_5\}$	3	$\{f_2,f_4,f_5\}$	3	$\{f_2,f_4,f_5\}$	3
Parity5+5	$\{f_2\text{-}f_4,f_6,f_8\}$ $\{f_3\text{-}f_7\}^{(1)}$	5	$\{f_2\text{-}f_4,f_6,f_8\}$ $\{f_3\text{-}f_7\}^{(1)}$	5	$\{f_2\text{-}f_4,f_6,f_8\}$	5
Parity5+2	$\{f_1,f_3\text{-}f_5,f_7\}^{(2)}$ $\{f_2\text{-}f_6\}^{(3)}$ $\{f_1\text{-}f_5\}^{(4)}$	5	$\{f_1,f_3\text{-}f_5,f_7\}^{(2)}$ $\{f_2\text{-}f_6\}^{(3)}$ $\{f_1\text{-}f_5\}^{(4)}$	5	$\{f_1\text{-}f_5\}$	5
Vote	$\{f_1\text{-}f_4,f_9,f_{11},f_{13},f_{15},f_{16}\}$	9	$\{f_1\text{-}f_4,f_9,f_{11},f_{13},f_{15},f_{16}\}$	9	$\{f_1\text{-}f_4,f_9,f_{11},f_{13},f_{15},f_{16}\}$	9
Lenses	$\{f_1\text{-}f_4\}$	4	$\{f_1\text{-}f_4\}$	4	$\{f_1\text{-}f_4\}$	4
Zoo	$\{f_3,f_4,f_6,f_8,f_{13}\}^{(1)}$ $\{f_3,f_4,f_6,f_9,f_{13}\}^{(2)}$ $\{f_3,f_6,f_8,f_{10},f_{13}\}^{(3)}$ $\{f_4,f_6,f_8,f_{12},f_{13}\}^{(4)}$ $\{f_3,f_6,f_8,f_{13},f_{16}\}^{(5)}$ $\{f_4,f_6,f_9,f_{12},f_{13}\}^{(6)}$ $\{f_3,f_6,f_9,f_{13},f_{16}\}^{(7)}$	5	$\{f_3,f_4,f_6,f_8,f_{13}\}^{(1)}$ $\{f_3,f_4,f_6,f_9,f_{13}\}^{(2)}$ $\{f_3,f_6,f_8,f_{10},f_{13}\}^{(3)}$ $\{f_4,f_6,f_8,f_{12},f_{13}\}^{(4)}$ $\{f_3,f_6,f_8,f_{13},f_{16}\}^{(5)}$ $\{f_4,f_6,f_9,f_{12},f_{13}\}^{(6)}$ $\{f_3,f_6,f_9,f_{13},f_{16}\}^{(7)}$	5	$\{f_3,f_4,f_6,f_8,f_{13}\}$	5
Mushroom	$\{f_5,f_{20},f_{21},f_{22}\}^{(1)}$ $\{f_4,f_5,f_{12},f_{22}\}^{(2)}$	4	$\{f_5,f_{20},f_{21},f_{22}\}^{(1)}$ $\{f_4,f_5,f_{12},f_{22}\}^{(2)}$	4	$\{f_5,f_{20},f_{21},f_{22}\}$	4
Sonar	758	4	661	4	$\{f_{12},f_{16},f_{20},f_{26}\}$	4

表 4 FOCUS 和 ABB 算法运算时间和搜索空间

Dataset	ABB t(ms)	Ev	FOCUS t(ms)	Ev
Corral	3	12	6	42
Monk1	19	20	10	24
Monk3	19	20	15	35
Parity5+5	406	112	636	518
Parity5+2	650	228	419	386
Vote	2697	908	48595	39967
Lenses	1	5	1	15
Zoo	132456	25344	883	4951
Mushroom	-	-	77032	5323
Sonar	-	-	42958	37485

Ev 为搜索空间大小。

表 5 BEMMIMAMI 算法运算时间 $t(ms)$

Dataset	BEMMIMAMI ($M=2p$)	BEMMIMAMI ($M=p$)	BEMMIMAMI ($M=1$)
Corral	3	3	3
Monk1	12	12	12
Monk3	12	12	12
Parity5+5	59	59	59
Parity5+2	76	76	47
Vote	203	203	78
Lenses	1	1	1
Zoo	669	110	16
Mushroom	93241	23672	7324
Sonar	423568	22781	344

表 4、表 5 和表 6 为它们的运算时间和搜索空间，从中可以看出 ABB 算法和 FOCUS 算法随着 p 的增大而急剧增大。对于 r 值较大情况，ABB 算法搜索空间比 FOCUS 小，因此这种情况 ABB 算法比较适用，反之 FOCUS 适用。相对于 ABB 和 FOCUS 算法，其搜索空间被大大压缩，运算时间被大大缩短，这是由于 BEMMIMAMI 算法是一个启发式算法。表 6 用 Ratio 显示了 BEMMIMAMI 算法相对于整个搜索空间的压缩率。可以看出，p 值越大，搜索空间压缩率越小。综合表 5 和表 6，M 值越大，搜索空间越大，运算时间越长。

表 7 为 BEMMIMAMI 算法属性约简结果的启发式粗糙集值约简分类错误率。可以看出，BEMMIMAMI 算法属性约简结果有良好的分类质量。

结论 通过从互信息角度分析，粗糙集理论的属性约简问题的实质就是要找到尽量小的初始属性集 F 的属性子集，使得其维持初始属性集的互信息。提出了基于 Beam 搜索的前向启发式粗糙集属性约简算法——BEMMIMAMI 算法。与一般的 Best-first 算法相比，它可以灵活地扩大搜索范围，得到更优的解；同时可以找到多个属性约简子集，这样可以得到更多的规则，因而可以提高识别率。

实验采用了 10 个标准的 UCI 数据集，从 4 个方面（约简、搜索空间大小和运行时间、分类错误率）来对 BEMMIMAMI 算法进行测试。属性集大小从 4 到 60，样本集大小从 24 到 8124。实验表明，BEMMIMAMI 粗糙集属性约简算法能够得到高质量的解（近似于最优解），特别是对于高维数据集。算法结果随着 Beam 宽度 M 的增大而趋于更优，但同时运算时间相应加长和搜索空间相应增大。因此在实际应用中，可以选择一个合适的 Beam 宽度，来保证运算时间和属性约简质量。

参 考 文 献

1 Pawlak Z. Rough Sets: Theoretical Aspects of Reasoning about Data [M]. Amsterdam: Kluwer Academic Publishers, 1991

2 Bazan J. A comparison of dynamic and non-dynamic rough set methods for extracting laws from decision system [A]. In: Polkowski L, Skowron A. Rough sets in Knowledge discovery [C]. Herdelberg: Phyisca-Verlag, 1998. 321～365

3 Narendra P, Fukunaga K. A branch and bound algorithm for feature subset selection [J]. IEEE Trans on Computer, 1977, 26

(9):917～922
4 Liu H, Motoda H. Feature selection for knowledge discovery and data mining [M]. Boston:Kluwer Acadcemic Press, 1998
5 苗多谦,王钰. 粗糙集理论中概念与运算的信息表示[J]. 软件学报, 1999, 10(2):113～116
6 王国胤,于洪,杨大春. 基于条件信息熵的决策表约简[J]. 计算机学报, 2002, 25(7):759～766
7 Aha D, Bankert R. A comparative evaluation of sequential feature selection algorithms [A]. In:Fisher D, Lenz H,eds. 5th International Workshop on Artificial Intelligence and Statistics [C]. New York: Springer, 1991
8 Cover T M. Elements of Information Theory [M]. New York: Wiley, 1991
9 苗夺谦,胡桂荣. 知识约简的一种启发式算法[J]. 计算机研究与发展, 1999, 36(6):681～684
10 王国胤. Rough 集理论与知识获取[M]. 西安:西安交通大学出版社,2001

表6 BEMMIMAMI 算法搜索空间大小

Dataset	All	BEMMIMAMI(M=2p)		BEMMIMAMI(M=p)		BEMMIMAMI(M=1)	
		Ev	*Ratio*	*Ev*	*Ratio*	*Ev*	*Ratio*
Corral	2^6	8	0.125	8	0.125	8	0.125
Monk1	2^6	8	0.125	8	0.125	8	0.125
Monk3	2^6	8	0.125	8	0.125	8	0.125
Parity5+5	2^{10}	12	0.012	12	0.012	12	0.012
Parity5+2	2^{10}	61	0.060	61	0.060	25	0.024
Vote	2^{16}	99	0.002	99	0.002	35	0.001
Lenses	2^4	6	0.375	6	0.375	6	0.375
Zoo	2^{16}	1751	0.027	406	0.006	57	0.001
Mushroom	2^{22}	4173	0.001	1366	0.000	106	0.000
Sonar	2^{60}	224583	0.000	10562	0.000	296	0.000

All 为整个搜索空间的大小，*Ratio* 为搜索空间压缩率，等于 *Ev* 除以 *All*。

表7 BEMMIMAMI 算法属性约简结果的启发式粗糙集值约简分类错误率

Dataset	BEMMIMAMI (M=2p)	BEMMIMAMI (M=p)	BEMMIMAMI (M=1)
Corral	0.00	0.00	0.00
Monk1	3.21	3.21	3.21
Monk3	1.45	1.45	1.45
Parity5+5	0.00	0.00	0.00
Parity5+2	$0.00^{(1)}$ $0.00^{(2)}$ $0.00^{(3)}$ $0.00^{(4)}$	$0.00^{(1)}$ $0.00^{(2)}$ $0.00^{(3)}$ $0.00^{(4)}$	0.00
Vote	5.23	5.23	5.23
Lenses	25.33	25.33	25.33
Zoo	$5.58^{(1)}$ $5.02^{(2)}$ $4.38^{(3)}$ $6.81^{(4)}$ $5.08^{(5)}$ $3.44^{(6)}$ $2.69^{(7)}$	$5.58^{(1)}$ $5.02^{(2)}$ $4.38^{(3)}$ $6.81^{(4)}$ $5.08^{(5)}$ $3.44^{(6)}$ $2.69^{(7)}$	5.58
Mushroom	$2.64^{(1)}$ $1.83^{(2)}$	$2.64^{(1)}$ $1.83^{(2)}$	2.64
Sonar	-	-	24.68

(上接第 185 页)

论了不同障碍对连通性的不同影响,提出了带障碍的分级聚类算法 OBHIEC。该算法首先依据障碍将整个区域划分成多个子区域,在每个子区域内不包含障碍,任意两个数据对象之间直接可达;然后在各个子区域内,使用不考虑障碍的聚类算法对子区域内的数据对象进行一级聚类,聚类后每个簇用其中心来代表,所有簇的代表点构成一个集合;最后在代表点集合上应用带障碍的聚类算法对代表点进行二级聚类,聚集为同一个簇的所有代表点所代表的数据对象同属一个聚类。分级聚类使得需要计算障碍距离的点对数目减少,并能处理数据分布密度不同的情况。本文实验结果表明,OBHIEC 算法能有效完成带障碍的聚类,并具有较好的增量特性,适用于大规模数据集。

参 考 文 献

1 Tung A K H, Hou Jean, Han Jiawei. Spatial Clustering in the Presence of Obstacles. Int conf on Data Engineering(ICDE'01), Heidelberg, Germany, April 2001
2 陈克平,等. 一种带障碍的网格弥散聚类算法 DCellO. 计算机研究与发展(增刊),2004,41(Suppl)
3 Han Jiawei, Kamber M. 数据挖掘——概念与技术(影印版). 北京: 机械工业出版社,2001
4 Ng R, Han J. Clarans: A method for clustering objects for spatial data mining. IEEE Transactions on Knowledge and Data Engineering, 2002, 14(5):1003～1016
5 Wang W, Yang J, Muntz R. STING: A statistical information grid approach to spatial data mining. In:Proc 1997 Int Conf Very Large Data Bases (VLDB'97), Athens, Greece, Aug 1997. 186～195
6 Karypis G, Han E-H, Kumar V. Chameleon: a hierarchical clustering algorithm using dynamic modeling. Computer, 1999, 32:32～68
7 Ng R, Han J. Efficient and effective clustering methods for spatial data mining. In. Bocca J, Jarke M, Zaniolo C, eds. Twentieth International Conference on Very Large Databases Santiago, Chile, 1994. 144～145