文章编号: 1002-0411(2001)04-313-05

一种基于小生境遗传算法的规则提取算法

陈建生　费　奇　陈学广

(华中科技大学系统工程研究所　武汉　430074)

摘　要: 本文描述一种基于小生境遗传算法的规则提取算法, 并从语言表述、规则编码、搜索策略三个方面做了讨论和分析. 相对于原有的一些方法, 本算法拥有适应性好、收敛快速的特点, 所生成的规则简洁, 易于理解.

关键词: 遗传算法, 归纳学习, 规则提取

中图分类号: $TP301$、$TP18$　　文献标识码: B

A NEW APPROACH TO RULE EXTRACTION BASED ON NICHING GENETIC ALGORITHM (RENGA)

CHEN Jian-sheng　FEI Qi　CHEN Xue-guang

(*Institute of System Science, Huazhong Univ. of Science and technology, Wuhan*　430074))

Abstract: In this paper, a approach to rule extraction based on niche genetic algorithm is proposed. The expression languages, encoding rules and searching strategy are discussed in detail. Compared with some other methods, this algorithm is more adaptive and more rapid to converge. The rules generated by this method are checked to be concise and easy to understand.

Keywords: genetic algorithm, inductive learning, rule extraction

1　引言(Introduction)

决策树归纳是归纳学习中的一个重要方法, 它以简明的思路、快捷的分类速度及较高的准确率获得了十分广泛的应用. 然而由于决策树归纳也有其固有的弱点, 近年来许多学者对其提出了改进方案. 这些工作大致可分为以下几类. 一类是在原有的算法上作出的修改[1], 但仍然属于经典决策树归纳的范畴; 另一类结合神经网络, 从经过训练的神经网络中抽取出决策树[2~4]; 第三类是用遗传算法进行决策规则的直接搜索[5, 6]. 本文所提出的是基于小生境遗传算法的改进算法, 属于上述的第三类.

本文在第二节介绍了决策树和小生境遗传算法(NGA), 第三节详细地给出了本文提出的算法, 第四节列出了一些试验与结果, 第五节是结束语.

[约定]　规则 R 一般都表示为合取范式: if $s(A_1)$ and $s(A_2)\cdots$ then class= c. 其中 $s(A_i)$ 是只含属性 A_i 的逻辑子句, 简称为子句. c 称为类别, 本文只讨论分为两类的例子集, 类别为 1 的称作正例集 P, 类别为 0 的称作反例集 N, 同时假定 P、N 分别有 p 个正例和 n 个反例. 一条规则被称作"好", 是指它覆盖正例且不覆盖任何反例. 显然, 我们寻找的目标规则 R_i 都是仅覆盖正例子的好规则. 假设 R_i 覆盖拥有 p_i 个正例的正例集 P_i. 本文的 *RENGA* 算法只讨论离散属性的情形, 对于连续属性的处理, 还有待以后改进.

2　决策树与小生境遗传算法(Introduction to decision tree and niche genetic algorithm)

[决策树]　决策树是一种递归地划分训练实例集的归纳学习方法. 它以简单、分类迅速、精确度较高的优点得到了广泛的应用.

但决策树归纳有其固有缺陷: (1) 每次选择单一属性用来分类, 缺乏灵活性; (2) 单纯用信息熵作分类准则, 没有考虑树的规模, 得到的规则不够简洁. 直接搜索最优化规则是解决这个问题的一个思路. 文献[1]指出, 在寻找最优化规则中有三个优化目标: (1) 生成最少数目的公式; (2) 生成最简洁的公式; (3) 生成只有最简公式的最优覆盖. 可惜的是, 这

收稿日期: 2000-10-20

几个问题都是 NP 难题. 目前只能使用启发式搜索算法.

[小生境遗传算法] 基本遗传算法在许多问题中都能够有效地搜索到全局最优点, 然而在面对要求同时搜索多个峰值的问题时, 基本遗传算法就显得无能为力. 小生境遗传算法正是被提出来解决这类问题, 它旨在限制相似个体的过多复制, 从而维持群体的多样性, 有效地对多个峰值搜索. 现代的小生境遗传算法主要分成两类, 排挤(Crowding)和共享(Sharing).

De Jong[8]在1975年提出基于排挤的小生境遗传算法. 其基本思想是: 在父代中随机选择 CF(Crowding Factor)个个体组成排挤成员, 然后用子代替换掉与其最相似的排挤成员中的个体. Mahfoud[9]指出这种方法不能很有效的识别多个锋值, 并提出了确定排挤(Deterministic Crowding)算法. 与 De Jong 不同, Mahfoud 将子代直接与它们的双亲比较, 并且在子代的适应值比它们的双亲更高时直接进行替换. 最近, Mengshoel[10]的可能性排挤(Probabilistic Crowding)改进了确定排挤, 它不直接将子代替换父代, 而是把子代与父代中相似的个体组成一个锦标赛(Tournament), 并根据适应值在锦标赛中引入概率.

另一类是基于共享的实现方法[11]. 其基本想法是引入共享函数来衡量个体之间相似程度, 再用共享函数来调整适应值. 当一个个体与其它个体较相似时, 其适应值会减小, 反之适应值会增大. 这样就限制了相似个体的过多复制, 从而造就了一种小生境的进化环境. 但合理的共享函数往往难以确定, 于是 Miller[12]于1996提出动态小生境共享(Dynamic Niche Sharing)算法, 以动态确定共享函数.

小生境遗传算法被认为是一种用于多峰值搜索的有效的方法.

3 RENGA 算法(the RENGA algorithm)

简单地应用遗传算法搜索规则空间往往效果并不理想, 或者耗时很长, 或者收敛于局部最小点, 或者没有找到全部规则. 前两个问题是由于规则平面非常不光滑造成的. 文[7]指出, 最优点周围有一些较差的点, 是遗传算法失效的重要原因. 第三个问题则是简单遗传算法的搜索策略的缺陷造成的.

为了解决这些困难, 本文算法从三方面做出了改进. 表述语言的扩充: 规则逻辑子句形式由 if $A_i = a_i$ 的点形式扩充为 if A_i in SCOPE$_i$ 的子集形式; 编码方法的改进: 由直接编码改进为例子组合编码; 搜索策略的改进: 由简单遗传算法改为用小生境遗传算法, 并且增加了新的遗传算子—— 规则之间的或运算. 这三个方面构成了 RENGA 算法的核心内容.

3.1 规则表述语言的扩充

语言表述决定了规则的表达形式. 语言的表达能力越差, 得到的规则就可能越复杂和难以理解; 相反地, 语言的表达能力越丰富, 得到的规则就越简洁和容易理解. 我们以规则(1): if $A_1 = 1$ and $A_2 = 2$ then class= 1; (2): if $A_1 \neq 3$ and $A_3 \neq 1$ then class= 1 张成的例子集为例, 来说明语言表述的差异. 其中属性 A_1, A_2, A_3 的取值范围分别是{1, 2, 3}, {1 , 2}, {1, 2, 3}.

[决策树表述语言] 这种表述语言有明显的弱点. 首先, 逻辑子句过于简单, 只利用了单一属性值; 第二, 如果某属性有 n 个属性值, 以此属性为节点就必然有 n 个分支, 多数时候这会产生繁杂的分支. 用决策树表达上面的例子可能会有7条规则. if $A_1 = 1$ and $A_2 = 2$ then class= 1; if $A_1 = 1$ and $A_2 = 1$ and $A_3 = 2$ then class= 1; if $A_1 = 1$ and $A_2 = 1$ and $A_3 = 3$ then class= 1; if $A_1 = 2$ and $A_2 = 1$ and $A_3 = 2$ then class= 1; if $A_1 = 2$ and $A_2 = 1$ and $A_3 = 3$ then class= 1; if $A_1 = 2$ and $A_2 = 2$ and $A_3 = 2$ then class = 1; if $A_1 = 2$ and $A_2 = 2$ and $A_3 = 3$ then class= 1; 除第一条规则外, A_2 的限定在其他规则中对于正确分类来说是多余的.

[简单的规则表述语言] 这种表述语言与决策树类似, 不同的是这里不存在节点概念, 因此规则表述相对自由一些. 同样用上面的例子, 可能得到的规则是: if $A_1 = 1$ and $A_2 = 2$ then class= 1; if $A_1 = 1$ and $A_3 = 2$ then class= 1; if $A_1 = 1$ and $A_3 = 3$ then class= 1; if $A_1 = 2$ and $A_3 = 2$ then class= 1; if $A_1 = 2$ and $A_3 = 3$ then class= 1; 规则数目少了两条.

[子集形式的规则表述语言] 子集形式的规则表述使用 if A_1 in SCOPE$_1$ and A_2 in SCOPE$_2$… then class= 1 的形式, SCOPE$_i$ 是属性 A_i 的值域的任意子集. 上面的例子可以表示为: if A_1 in {1} and A_2 in {2} then class= 1; if A_1 in {1, 2} and A_2 in {2, 3} then class= 1; 可见子集形式比前面两种表述方式更简洁和更容易理解. 另外容易看出, 如果 SCOPE$_i$ 是属性 A_i 某值 a 的补集, 则子句 if A_i in SCOPE$_i$ 等价于 if $A_i = a$ 子句的非逻辑. 也就是说这种规则表示方法可以表达单一子句的非逻辑, 这是

前面两种方式不能表达的.

子集形式的规则表述拥有比决策树更丰富、更灵活的语言, 能够表示更高层次的规则, 得到的规则数目也是最少的. 本文的 *RENGA* 算法采用子集形式的规则表述语言.

3.2 例子组合编码

编码是应用遗传算法时要解决的首要问题, 也是设计遗传算法时的一个关键步骤. De Jong 曾提出了两条操作性较强的实用编码原则. 原则一(有意义积木块编码原则): 应使用能易于产生与所求问题相关的且具有低阶、短定义长度模式的编码方案. 原则二: 应使用能使问题得到自然表示的具有最小编码字符集的编码方案. 最容易想到的就是直接编码, 而本文则采用例子组合编码.

[直接编码] 是将每个子句 if A_i in SCOPE$_i$ 或 if $A_i = a$ 编码为二进制串, 然后把多个子句编码连结在一起成为一个染色体. 正如本节序言中提到, 这种编码方式会造成规则平面不光滑, 对搜索十分不利. 而且 if A_i in SCOPE$_i$ 难以编码, 不是规则的自然表示.

[例子组合编码] 例子组合编码的二进制位数与正例子个数相等的二进制串, 二进制串的第 i 位对应着第 i 正例子, "1"表示编码包含该正例子, "0"表示不包含. 规则中的每个 SCOPE$_i$ 可以由编码从如下过程得到: 将编码中"1"对应的所有正例子取出, 第 i 个属性 A_i 在这些正例子中出现的所有属性值组成的集合就构成 SCOPE$_i$. 结合全部这样的 SCOPE$_i$ 就是编码所表达的规则 if A_1 in SCOPE$_1$ and A_2 in SCOPE$_2$… then class= 1.

我们称那些仅仅覆盖 P_i 的子集的规则为 R_i 的组合积木块. 例子组合编码有以下几个优点:

(1) 例子组合编码是子句 if A_i in SCOPE$_i$ 的自然表示. (2)例子组合编码产生的规则肯定覆盖一些正例子; 而直接编码却会产生不覆盖任何正例子的无用规则, 对搜索过程不利. (3)在随机搜索过程中, 组合积木块被搜索到的概率是 $2^{p_i}/2^p$, 可见 R_i 覆盖正例越多(p_i 越大), 其组合积木块被搜索到的概率越大. 而直接编码在随机搜索中, 所有的规则不论好坏被搜索到的概率是一样的. (4)目标规则 R_i 的组合积木块之间的或运算肯定生成更接近 R_i 的适应值更高的好规则, 大大加快搜索过程.

但是, 例子组合编码的编码空间大小是 2^p, 直接编码的编码空间与规则空间的大小是一致的. 当正例集非常大时, 例子组合编码比直接编码的编码空间大得多. 编码空间太大, 搜索过程可能会十分慢, 但从试验结果看出: 当正例集拥有小或中等规模时, 例子组合编码的搜索性能优异, 收敛十分快速.

3.3 或运算

或运算是我们为了与例子组合编码相适应而新设计的遗传算子, 它是规则编码之间的按位或运算, 用符号"|"表示. 定义为: $w(u, v) = u | v$, 其中 u, v 是两个个体编码.

如果目标规则 R_i 覆盖正例集 P_i, u、v 是 R_i 组合积木块并且覆盖分别有 p_u、p_v 个正例的正例集 P_u、P_v, 由组合积木块的定义知 P_u、$P_v \subset P_i$. 那么 $w = u | v$ 将会覆盖集合 $P_w = P_u \cup P_v$, 则规则 w 比 u、v 覆盖更多的正例、更接近目标规则 R_i, 也有更高的适应值.

换句话说, 积木块间的或运算肯定生成更接近 Ri 的适应值更高的好规则, 这样就极大地加快目标规则的搜索. 或运算的作用是十分明显的, 可以说本算法的快速收敛特性大部分的功劳应归于积木块之间的或运算. 需要说明的是: 不是所有规则间的或运算都能生成更好的规则, 只有同一条目标规则的积木块之间的或运算才肯定生成更好的规则.

RENGA 算法在增加编码之间或运算的同时, 把传统的遗传算子"交叉"删除, 因为在本算法中或运算已经取代了"交叉"运算的信息交换的作用, 并且比"交叉"运算更有效, 删除"交叉"使 RENGA 算法更简洁.

3.4 适应值的确定

为了满足最优覆盖的目标—— 生成最少数目的公式, 必须要求目标规则尽可能多地覆盖正例子. 因此规则的适应值, 可以简便地作如下定义. 设规则 R_i 覆盖 p_i 个正例、n_i 个反例. 适应值 $\mathrm{fitness}(R_i) = \begin{cases} p_i & \text{if } n_i = 0 \\ 0 & \text{if } n_i \neq 0 \end{cases}$. 当 R_i 为好规则时, 所覆盖正例数 p_i 越大越好; 当 R_i 不是好规则时, 适应值定为零, 这条规则在下一代遗传中就会被淘汰.

3.5 应用小生境遗传算法[7]

规则提取是典型的多峰值函数搜索问题, 所以使用小生境遗传算法正好合适. 本文用蕴含关系来衡量个体之间的相似程度, 并且为了简单起见, 只考虑了"蕴含"与"非蕴含"两种状态. 具体如下: 首先将群体按适应度进行降序排序, 从高适应度到低适应度顺序比较各个体之间的规则蕴含关系, 被蕴含的个体马上被淘汰. 因为存在蕴含关系的规则之间的交叉或组合运算不可能产生适应值更高的个体, 所

以将其淘汰是合理的. 然后随机增加新个体, 以维持群体大小的恒定. 这样, 一条最优规则附近只存在一个优良的个体, 从而既维护了群体的多样性, 又使得个体之间彼此分开, 这样就实现了小生境遗传算法.

4 算法步骤及试验结果(Procedure and test results)

[算法步骤]

(1) 从正例集中随机选取 M 个个体组成初始群体.

(2) 将各个个体按适应度进行降序排序, 记忆前 N 个个体($N < M$).

(3) 对群体进行或、选择和变异等运算.

(4) 小生境淘汰运算. 将第三步得到的 M 个个体和第二步所记忆的 N 个个体合在一起, 得到一个含有 $M+N$ 个个体的新群体; 对这 $M+N$ 个个体按适应度进行降序排序, 按适应值从高到低比较个体间的规则蕴含关系, 淘汰被蕴含的个体.

(5) 如果剩余个体数量小于 M, 则从正例集中随机选择适当数目的个体, 使群体的大小恒定.

(6) 终止条件判断. 若不满足终止条件, 则: 将第五步所到群体中适应值最高的前 M 个个体作为新一代群体. 然后转向第二步. 若满足终止条件, 转向第七步.

(7) 从最高适应值的个体开始逐个输出结果, 直到输出集已经覆盖所有实例为止.

(8) 在综合考虑规则数目与精度后, 将适应值最低的几个个体删除. 如果规则数目不多, 也可以不删除.

(9) 将输出的个体解码为规则集, 然后将所得规则集泛化(一般化), 即: 使规则中某逻辑子句变成全称子句. 如果泛化后的规则有效(不覆盖任何反例), 则用新规则替代旧规则. 重复此步骤, 直到没有规则可以泛化为止. 输出最终结果.

即使第九步不进行泛化, 理论上已经得到的已经是最优覆盖规则. 泛化的目的是使规则更简洁和更容易理解, 即为了满足第二节所述的第二个优化目标.

[试验用例介绍]

本文所举的例子一、二及其 C4.5 算法的结果来源于 http://yoda.cis.temple.edu:8080/cgi-bin/c45/nph-c45. 第一个例子由以下两条规则张成: (1) if $A_5=3$ and $A_4=1$ then class=1. (2) if $A_5 \neq 4$ and $A_2 \neq 3$ then class=1, 共有 122 个例子, 60 个正例. 原例有 5% 的噪音, 本文将噪音滤掉得到本文的测试用例. 第二个例子由下面两条规则张成: (1) if $A_1=A_2$ then class=1. (2) if $A_5=1$ then class=1, 124 个例子, 62 个正例. 例子三是常用的 The Wisconsin breast cancer data set, 有 9 个输入, 两个输出, 去噪后共有 683 个实例, 其中 444 个正例, 239 个反例. 这些实例 300 个被用作训练, 383 个用作测试. 表 1 中例子三的部分数据来源于文献[3].

表 1 试验结果及分析

Tab. 1 Test Results and Analysis

例子数(正例数: 反例数)	算法	进化代数	节点数*	树叶数*	精度
例子一 122 (60: 62)	C4.5		25/12(剪枝前/后)	17/9(剪枝前/后)	96.7%/93.4%
	RENGA	10	5	3	100%**
例子二 124 (62: 62)	C4.5		43/18(剪枝前/后)	28/12(剪枝前/后)	90.3%/83.9%
	RENGA	5	8	5	100%**
例子三 683 (444: 239)	ID3		39	20	96.8%
	文献[3]的算法		13	7	95.91%
			3	2	94.74%
	RENGA	8	9	2	95.90%

* 为便于比较, 本算法中节点数和树叶数已按决策树的形式经过转换.

** 算法找到的规则与目标规则一致, 所以精度为 100% .

结果是十分明显的, 本文的 RENGA 算法在例子一、例子二中完全准确找到目标规则. 例子 1 中, 本算法成功发现非逻辑. 在例子三中, 与 ID3 比较, 虽然精度稍差, 但所得到的规则却简洁得多. 与文献[3]基于神经网络(其中也用到遗传算法)的方法比较, 在精度、规模相若的情形下, 本文算法拥有简单、实现容易、运算速度较神经网络快的优点.

本文的算法相对于经典决策树和简单遗传算法

有一定的优势. 非逻辑的获得得益于语言表述的扩展, 收敛速度得益于编码方法和规则间的或运算, 而小生境遗传算法则保证了规则对例子的有效覆盖. 但是本算法比决策树归纳耗时长, 这不是因为遗传算法进化代数多, 而是算法本身计算量较大的缘故.

5　结束语(Conclusion)

本文提出的算法的应用效果比较理想. 但是, 连续属性的处理, 噪音数据的处理, 算法收敛速度的研究和如何应用到多类别分类问题等也有待继续深入研究.

参　考　文　献(References)

1　洪家荣等. 一种新的决策树归纳学习算法. 计算机学报, 1995, (7)

2　Sethi I K, Yoo Y H. Structure-driven Induction of Decision Tree ClassiFiers Through Neural Learning, Pattern Recognition, 1997, **30**(11): 1893

3　R Krishnan, G Sivakumar, P Bhattacharya. Extracting Decision Trees From Trained Neural Networks, Pattern Recognition, 1999, **32**(12)

4　Schmitz, Greogor P J, Aldrich, Chris, ANN-DT: An Algorithm for Extraction of Decision Trees From Artificial Neural Networks., IEEE Transactions on Neural Networks 1999, **10**(6)

5　肖　勇, 陈意云. 用遗传算法构造决策树. 计算机研究与发展, 1998, (1)

6　张雪江. 利用基因遗传算法从数据库自动生成知识库. 计算机工程与设计, 1997, **18**(6)

7　周　明. 遗传算法原理及应用. 国防工业出版社, 1999

8　De Jong, K A. An Analysis of the Behavior of a Class of Genetic Adaptive Systems. Ph. D. Dissertation, University of Michigan, 1975, 76~ 9381

9　Mahfoud S W. Crowding and Preselection Revisited. In Manner, R., & Manderick, B. (Eds.), Parallel Problem Solving from Nature, 2. Amsterdam, The Netherlands: Elsevier Science., 1992, 27~ 36

10　Mengshoel O J, Goldberg D E. Probabilistic Crowding: Deterministic Crowding with Probabilistic Replacement. In Banzhaf, W. et al., Proceedings of the Genetic and Evolutionary Computation Conference 1999 (GECCO- 99). San Fransisco, CA: Morgan Kaufmann., 1999

11　Goldberg D E, Richardson J. Genetic Algorithms with Sharing for Multimodal Function Ptimization. In Grefenstette, J. J. (Ed.), Proceedings of the Second International Conference on Genetic lgorithms. Hillsdale, NJ: Lawrence Erlbaum Associates., 1987, 41~ 49

12　Miller B L, Shaw M J. Genetic Algorithms with Dynamic Niche Sharing for Multimodal Function Optimization In IEEE International Conference on Evolutionary Computation, Piscataway, NJ: IEEE Press, 1996, 786~ 791

作者简介

陈建生(1974-), 男, 研究生. 研究领域为神经网络和知识发现的研究工作.