第30卷 第9期 计 算 机 工 程 2004年5月
Vol.30 №9 Computer Engineering May 2004
·软件技术与数据库· 文章编号：1000—3428(2004)09—0094—03 文献标识码：A 中图分类号：TP301.6

决策树分类技术研究

栾丽华，吉根林
（南京师范大学计算机系，南京 210097）

摘 要：决策树分类是一种重要的数据分类技术。ID3、C4.5和EC4.5是建立决策树的常用算法，但目前国内对一些新的决策树分类算法研究较少。为此，在消化大量文献资料的基础上，研究了CART、SLIQ、SPRINT、PUBLIC等新算法，对各种决策树分类算法的基本思想进行阐述，并分析比较了各种算法的主要特性，为数据分类研究者提供借鉴。

关键词：决策树；CART；SLIQ；SPRINT；PUBLIC

The Study on Decision Tree Classification Techniques

LUAN Lihua，JI Genlin
(Computer Department of Nanjing Normal University, Nanjing 210097)

【**Abstract**】 Decision tree is one of the most important data classification techniques. Algorithms ID3, C4.5 and EC4.5 are widely used to construct decision trees, but little work on new decision tree algorithms has been done. Some new decision tree algorithms are studied, including CART, SLIQ, SPRINT and PUBLIC. The basic ideas and main features of these algorithms are discussed in this paper.

【**Key words**】 Decision tree；CART； SLIQ；SPRINT；PUBLIC

数据分类是数据挖掘中的一种分析方法，通过学习训练集构造一个分类函数或分类模型，该函数或模型能够把数据记录映射到给定类别中的某一个，从而可以应用于数据预测。训练集由一组数据库记录构成，记录形式可以表示为（$V_1,V_2,\cdots,V_n$, C），其中V_i表示样本的属性值，C表示类别。分类模型的构造方法包括统计方法（如贝叶斯方法）、机器学习方法（如决策树方法）、神经网络方法、粗集方法和遗传算法等。

本文阐述了各种决策树分类算法的基本思想,包括ID3、C4.5、CART、SLIQ、SPRINT和PUBLIC，通过分析比较，总结了各种算法的主要特性，为国内研究者提供借鉴。

1 决策树分类算法

1.1 ID3算法

早期决策树算法是1986年由Quilan提出的ID3算法[1]，它是一个从上到下、分而治之的归纳过程。ID3算法选择具有最高信息增益的属性作为测试属性。

定义1：设样本集T按离散属性A的s个不同的取值，划分为T_1，…,T_s共s个子集，则T用A进行划分的信息增益为：

$gain(A,T)=\inf(T)-\sum_{i=1}^{s}\frac{|T_i|}{|T|}\times\inf(T_i)$，其中$\inf(T)$表示T的信息熵。设T中有m个类，则$\inf(T)=-\sum_{j=1}^{m}p_j\times\log_2(p_j)$，其中，$p_j$表示T中包含类j的概率。

ID3算法思想描述如下：

(1) 初始化决策树T为只含一个树根(X,Q)，其中X是全体样本集，Q为全体属性集。

(2) if(T中所有叶节点(X’,Q’)都满足X属于同一类或Q’为空) then 算法停止；

(3) else

{ 任取一个不具有(2)中所述状态的叶节点(X’,Q’)；

(4) for each Q’中的属性A do 计算信息增益gain(A, X’)；

(5) 选择具有最高信息增益的属性B作为节点(X’, Q’)的测试属性；

(6) for each B的取值b_i do

{从该节点(X’, Q’)伸出分支，代表测试输出B=b_i；

求得X中B值等于b_i的子集X_i, 并生成相应的叶节点 (X_i’, Q’-{B})；}

(7) 转(2)；}

1.2 C4.5算法

ID3算法不能处理连续属性,且构造的决策树过度适合[1]数据。对此，C4.5算法[2]作了两点改进。C4.5算法挑选具有最高信息增益率的属性作为测试属性。

定义2：设样本集T按离散属性A的s个不同的取值，划分为T_1，…,T_s共s个子集，则用A对T进行划分的信息增益率为 $ratio(A,T)=gain(A,T)/split(A,T)$，其中

$split(A,T)=-\sum_{i=1}^{s}\frac{|T_i|}{|T|}\times\log_2(\frac{|T_i|}{|T|})$。

C4.5算法不仅可以处理离散属性(方法与ID3中相同)，还可以处理连续属性。它规定在连续属性A上的测试导致两个分支,分别对应于条件A≤V和A＞V，其中V称作局部阈值。若A是测试属性，则V称作阈值。要确定A的局部阈值，首先对T中A属性值已知的样本进行快速排序，依次考察排序后的每对相邻值的中间值v，以及对应的划分条件A≤v和A＞v。假定样本中A有m个不同取值，则存在m-1个中间值v，分别对应m-1个可能的信息增益率ratiov'。若ratiov'值最大，则v’就是A的局部阈值，$ratio_{v'}$就是A的信息增益率。当考察完所有属性，就可以得到N的测试属性X。若X是连续的，则须在整个训练集上线性搜索X值，找到从低到高最接近于X的局部阈值的X值作为阈值。

生成决策树后，C4.5算法采取剪枝技术来纠正过度适合问题，即剪去树中不能提高预测准确率的分支。若分支节点

基金项目：江苏省教育厅自然科学基金资助项目(2001SXXTSJB12)
作者简介：栾丽华（1980—），女，硕士生，研究方向为数据挖掘技术；吉根林，副教授
收稿日期：2003-04-01 E-mail：yimu_njnu@163.com

N的分类错误多于将N中所有样本归为一类而导致的分类错误，则说明N无须划分，因而将节点N的孩子剪去。

1.3 EC4.5算法

为了减少C4.5为连续型测试属性线性搜索阈值而付出的代价，EC4.5算法[2]采用二分搜索取代线性搜索。EC4.5还提出3种不同的计算信息增益与阈值的改进策略，可以根据实际情况采取其中最佳的一种。但EC4.5占用内存比C4.5多。

1.4 CART算法

CART[3]是Classification And Regression Tree的简称，可以处理高度倾斜或多态的数值型数据，也可处理顺序或无序的类属型数据。CART选择具有最小gini系数值的属性作为测试属性，gini值越小，样本的“纯净度”越高，划分效果越好。

定义3：设样本集T中包含n个类，则 $gini(T)=1-\sum_{j=1}^{n}p_j^2$，其中$p_j$是T中包含类j的概率。若将T划分为两个子集$T_1$和$T_2$，则 $gini(T_1,T_2)=\frac{|T_1|}{|T|}gini(T_1)+\frac{|T_2|}{|T|}gini(T_2)$。

与C4.5算法类似，CART算法也是先建树后剪枝，但在具体实现上有所不同。由于二叉树不易产生数据碎片，精确度往往高于多叉树[1]，因此CART算法采用2分递归划分，在分支节点上进行布尔测试，判断条件为真的划归左分支，否则划归右分支，最终形成一棵二叉决策树。对于连续属性A，判断A≤V是否成立(同C4.5算法)；对于离散型属性A，判断A∈S’是否成立，其中S’是属性A所有取值的子集，可用贪心算法或穷举法确定，参见文献[4]。

CART算法在建树时，不管节点N是否将被划分，均给N标记相应的类，方法是判断不等式 $\frac{C(j|i)\prod(i)N_i(t)}{C(i|j)\prod(j)N_j(t)}>\frac{N_i}{N_j}$，若对于除i以外的所有类j都成立，则将N标记为类i。其中，Π(i)表示类i的先验概率，N_i是训练集中类i的数量，$N_i(t)$是节点N的样本中类i的数量。C(j|i)表示将i错误分类为j的代价，可通过查找决策损失(代价)矩阵得到。

CART算法在满足下述条件之一时停止建树：(1)所有叶节点中的样本数为1或者样本属于同一类，(2)决策树高度到达用户设置的阈值。

CART算法在剪枝阶段，采用代价-复杂性剪枝算法。决策树复杂度与分类精确度之间的关系如图1所示[3]。图1说明当决策树复杂度超过一定程度后，随着复杂度的提高，测试集的分类精确度反而会降低。因此，建立的决策树不宜太复杂，需进行剪枝。该剪枝算法依赖于复杂性参数α(α随树复杂度的增加而减小)，当增加一个节点引起的分类精确度变化量小于树复杂度变化的α倍时，则须剪去该节点。故建立一棵既能精确分类，又不过度适合的决策树的关键是求解一个合适的α值。

CART采用交叉认证法来求解最优树，即将训练集随机分成N个集合(假定每个集合的数据分布近似或相同)，其中，一个集合留作独立的测试集，其余N-1个集合合并后作为训练集。这样的组合方案有N种，分别对应N次完整的建树剪枝过程、以及生成的N个不同的“最佳树模型”。事实表明，N个“最佳模型”的平均性能非常近似于由整个训练集得出的原始模型在独立测试集上的性能，所以该法可避免使用独立测试集。

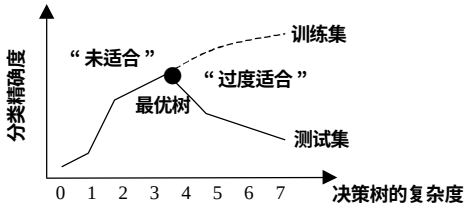

图1 决策树的复杂度与分类精确度之间的关系

1.5 SLIQ算法

上述算法由于要求训练样本驻留内存，因此不适合处理大规模数据。为此，IBM研究人员提出了一种快速的、可伸缩的、适合处理较大规模数据的决策树分类算法SLIQ[4](Supervised Learning In Quest)。该算法利用3种数据结构来构造树，分别是属性表、类表和类直方图。属性表含有两个字段：属性值和样本号。类表也含有两个字段：样本类别和样本所属叶节点。类表的第k条记录对应于训练集中第k个样本(样本号为k)，所以属性表和类表之间可以建立关联。类表可以随时指示样本所属的划分，所以必须长驻内存。每个属性都有一张属性表，可以驻留磁盘。类直方图附属在叶节点上，用来描述节点上某个属性的类别分布。描述连续属性分布时，它由一组二元组<类别，该类别的样本数>组成；描述离散属性分布时，它由一组三元组<属性值，类别，该类别中取该属性值的样本数>组成。随着算法的执行，类直方图中的值不断更新。

SLIQ算法在建树阶段，对连续属性采取预排序技术与广度优先相结合的策略生成树，对离散属性采取快速的求子集算法确定划分条件。具体步骤如下：

(1) 建立类表和各个属性表，并且进行预排序，即对每个连续属性的属性表进行独立排序，以避免在每个节点都要给连续属性值重新排序。

(2) 如果每个叶节点中的样本都能归成一类，则算法停止；否则转(3)。

(3) 利用属性表寻找拥有最小gini值的划分作为最佳划分方案。算法一次只处理一张属性表(假设对应于属性A)，从上往下每读一条记录，就根据样本号关联到类表的相关记录，找到样本所在的叶节点，从而更新叶节点上的类直方图，若A是连续的，则还要根据A的当前取值v计算对应于判断A≤v的gini值。若A是离散的，则在扫描完A属性表后，用贪心算法或穷举法计算最佳的S’。当扫描结束时，就可确定在各叶节点上依据属性A进行划分的最佳方案，各叶节点上的划分方案不一定相同。同理，当扫描完所有属性表时，可确定当前树中所有叶节点的最佳划分。

(4) 根据(3)步得到的最佳方案划分节点，判断为真的样本划归左孩子节点，否则划归右孩子节点。这样，(3)(4)步就构成了广度优先的生成树策略。

(5) 更新类表中的第二项，使之指向样本划分后所在的叶节点。

(6) 转(2)。

SLIQ算法采取基于MDL(最小描述长度)原则进行剪枝，即根据决策树的编码代价的大小进行剪枝。剪枝的目标就是寻找最小代价树。

定义4：决策树编码代价 =树结构的编码代价 +在内部节点上进行测试的编码代价 +训练集的编码代价。

树结构的编码代价等于树中各节点t的编码代价L(t)之和，而L(t)按如下情况进行定义：(1)若树中只含有叶节点或度为2的节点，只需1 bit区分节点，则L(t)为1；(2)若树中含有叶节点、只有左孩子的节点、只有右孩子的节点及度为2的节点，需2 bit区分节点，则L(t)为2；(3)若树中仅考虑分支

节点，即只有左孩子的节点、只有右孩子的节点及度为2的节点，需log3个bit区分节点，则L(t)为log3。

在内部节点上进行测试的编码代价L_{test}与测试属性的类型有关。若测试属性是连续属性，则L_{test}定义为1；若测试属性是离散属性A，设在决策树中用A进行测试的次数为S，则L_{test}定义为lnS。

训练集的编码代价定义为给训练集分类得到的所有分类错误之和Errors。

根据上述定义，每个节点t的编码代价C(t)的计算方法如下：

(1) 如果t为叶节点，则C(t)= L(t)+ Errors(t)

(2) 如果t含有两个孩子t_1、t_2，则C(t)= L(t)+ L_{test}+ C(t_1)+ C(t_2)

(3) 如果t只有左孩子t_1，则C(t)= L(t)+ L_{test} + C(t_1)+ C’(t_2)

(4) 如果t只有右孩子t_2，则C(t)= L(t)+ L_{test} + C(t_2)+ C’(t_1)

剪枝一般有3种策略，分别是完全剪枝、部分剪枝和混合剪枝。对于某个分支节点，完全剪枝只考虑上述(1)(2)情况，即要么不剪枝，要么将该节点的两个孩子全部剪去；部分剪枝考虑上述4种情况，从而决定不剪枝、剪去节点的两个孩子、还是只剪去其中一个孩子，其中(3)(4)中的C’(t_i)表示被剪枝孩子中的样本留在父节点t中导致的分类错误；混合策略分别使用前两种策略进行分步剪枝。选定一种剪枝策略后,从叶节点向上,为每个节点t计算不同情况时的C (t)，再根据最小的C(t)所对应的情况进行剪枝。

实践证明，对于前面算法可以处理的小规模训练集，SLIQ的运行速度更快，生成的决策树更小，预测的精确度较高；对于前面算法无法处理的大型训练集，SILQ精确度更高，优势更明显。

1.6 SPRINT算法

SLIQ算法要求类表驻留内存。当训练集增加导致类表放不进内存时，算法就无法进行，这限制了SLIQ处理数据的最大规模。为此，IBM研究人员提出可伸缩、可并行化的决策树算法SPRINT[5](Scalable PaRallelizable INndution of decision Tree)，它消除了所有内存限制，运行速度快，且允许多个处理器协同创建一个决策树模型。SPRINT定义了两种数据结构，分别是属性表和直方图。属性表由属性值、类别属性和样本号3个字段组成，它随节点的扩展而划分，并附属于相应的子节点。直方图附属在节点上，用来描述节点上某个属性的类别分布。当描述连续属性的类分布时，节点上关联两个直方图C_{below}和C_{above}，前者描述已处理样本的类别分布，后者描述未处理样本的类别分布，两者的值皆随算法进行而更新；当描述离散属性的类分布时，节点上只关联一个直方图count matrix。

与SLIQ算法不同，SPRINT算法采取传统的深度优先生成树策略，具体步骤如下：

(1) 生成根节点，并为所有属性建立属性表，同时预排序连续属性的属性表。

(2) 如果节点中的样本可归成一类，则算法停止；否则转(3)。

(3) 利用属性表寻找拥有最小gini值的划分作为最佳划分方案。算法依次扫描该节点上的每张属性表。由于计算gini值需要该节点上直方图的信息，因此处理每一张属性表前均要清空并初始化直方图。对于连续属性A，要求初始化C_{below}为0、C_{above}为该节点上的样本关于A的总体分布。每扫描A属性表的一条记录(设属性值为v)，都要更新一次C_{below}和C_{above},且计算对应于划分A≤v的gini值。这样，一遍扫描完A属性表，就可得到该节点上关于A属性的最佳划分，并记录下来。对于离散属性B，先清空count matrix，在扫描B属性表的同时更新count matrix，扫描完后，根据构造好的count matrix求解最佳划分中的S，并记录下来。当扫描完该节点的所有属性表时，就可得到该节点的最佳划分方案。

(4) 根据划分方案，生成该节点的两个子节点N1、N2。

(5)划分该节点上的各属性表，使之关联到N1或N2上。首先划分测试属性的属性表：依次扫描记录<属性值x，类别属性c，样本号id>，判断x属于哪个孩子，然后将该条记录移至该孩子节点的新属性表中，同时将id值写入Hash表，以保存划分后样本所在孩子节点的信息。然后划分其余属性表：依次扫描记录，依据样本号查找Hash表中该样本划分后所在的孩子节点，然后将该记录移至该孩子节点的新属性表中。

(6)分别转(2)考察N1 、N2节点。

SPRINT算法在剪枝阶段进行基于MDL的剪枝，至此构成了SPRINT的串行化算法。将串行化算法稍加改进，就成为并行化算法：将训练集平均分布到N个处理器上，而后每个处理器生成自己的数据分片。由于连续属性值要求全局排序，因此要将N个处理器上的连续属性的属性表综合重排序，再平均分布到N个处理器上。在建立Hash表之前，需要从N个处理器上收集所有的样本号信息。

实践表明，在SLIQ的类表可以存进内存的情况下，SPRINT比SLIQ执行得慢；然而在训练集规模超过SLIQ能承受的最大规模后，SPRINT的效率比SLIQ的要好得多，且数据量越大，SPRINT的效率越高！

1.7 PUBLIC算法

上述含有剪枝的算法都是分成两步进行，即先建树再剪枝。PUBLIC[6] (Pruning and Building Integrated in Classification)算法将建树、剪枝结合到一步完成，在建树阶段不生成会被剪枝的子树，因而大大提高了效率。

PUBLIC的建树基于SPRINT方法、剪枝基于MDL原则，但MDL原则不能直接应用。因为生长不完全的树提供的信息不完全，所以对于仍要扩展的叶节点，估算的编码代价偏高，这会导致过度剪枝，从而降低预测精确度。PUBLIC采用低估策略来矫正过高的代价估算，即对将要扩展的叶节点计算编码代价的较低阈值，而对于另两类叶节点(剪枝形成的、不能被扩展的)，估算方法不变。计算较低阈值的方法有多种，但最简单有效的方法是将它设置为1。另外，PUBLIC对决策树编码代价的计算方法不同于SLIQ。PUBLIC对节点N进行判断剪枝的步骤如下：

(1) if (N是将被扩展的叶节点) then 返回N的较低阈值cost1；

(2) if (N是剪枝形成的或不可扩展的叶节点) then 返回N的编码代价cost2；

(3) else

{计算N的编码代价cost3；

//N是含有两个子节点N1和N2的分支节点

(4) if(cost2<cost3) then{将N1和N2剪枝;N变为剪枝后的叶节点;}

(5) 返回cost2与cost3中的较小值；}

PUBLIC在建树过程中定期调用剪枝算法，避免生成被剪枝的节点，因而节省了开销，提高了效率。

2 结束语

本文论述了7种决策树算法。在构造决策树过程中，选择测试属性的启发信息有信息增益、信息增益率、gini系数；决策树的结构有多分支、两分支；剪枝方法有基于分类错误的剪枝、基于代价复杂性的剪枝、基于MDL原理的剪枝。这些决策树算法中，有的只建树不剪枝，有的先建树后剪枝，还有的将建树和剪枝集成在一起。预测准确率、速度、鲁棒性、可伸缩性是决策树算法的主要性能指标，围绕这些性能我们还可以对决策树算法进行改进，以适应不同的分类要求。 （下转第105页）

我们自己设计的IPv6路由器，以可编程的高速网络处理器np3400作为核心处理器，保证线速处理。

IPv6路由器的内部结构如图3所示。包括3个部分：网络处理器，CPU和交换结构。网络处理器完成数据平面的功能，实现数据报的实时处理、业务流表的查询、路由转发等；CPU 完成控制平面的功能，实现协议处理、业务流维护和网络管理等；交换结构实现数据报的调度和高速转发。

3.2 实现方法

为了提高IPv6路由器的报文处理速度，减小CPU的负荷，将6 to 4 tunnel在np3400的微码层实现。一个运行6 to 4 tunnel的IPv6路由器需要同时具备将IPv6数据报封装成IPv4 数据报和将封装的IPv6数据报拆封的能力。为了说明的方便，将路由器上连接IPv6网络的端口称为下联端口，连接IPv4网络的端口称为上联端口。下联端口进行IPv6报文封装，上联端口进行IPv6-in-IPv4报文的解封装。

(1) 报文封装的实现。当路由器的下联端口接收到一个IPv6报文之后，对于需要tunnel的报文，建立两种机制进行判断：第1种是判断目的地址前缀是否为2 002，若是，进行tunnel处理，否则将该报文送到普通报文处理代码段进行处理；第2种是上层软件在该报文的路由查找表项中设置相应的tunnel标志位，标志使能，进行tunnel处理，否则作为普通报文进行处理。

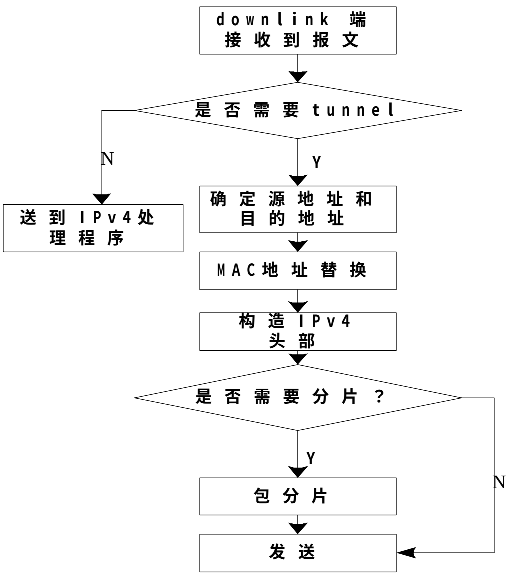

图4 报文封装流程图

报文进入tunnel封装程序之后，程序根据IPv6的源和目的地址确定tunnel的源和目的IPv4地址，进行MAC地址替换，构建IPv4头部，判断报文是否需要分段，根据需要将报文适当分段，封装报文，将报文转发到目的端口。报文封装具体的程序流程如图4。

(2) 报文拆封的实现。在隧道的出口处，被封装的报文首先出现在路由器的IPv4程序段。路由器的上联端口接到一个IPv4报文，首先查看它的协议字段，若该字段为41，目的地址与路由器自己的IPv4地址相同，表明该报文封装了一个IPv6报文，并且隧道的出口端就是路由器自己，所以该报文需要送到tunnel拆封程序处理[4]，否则，将该报文送到普通IPv4报文处理程序。拆封程序首先判断接收到的报文是否被分片过，如果该报文是一个报文的分片，路由器继续接收其余的报文分片。当接收到全部分片后，路由器对分片进行报文重组。移去IPv4头部，将IPv6首部的跳极限字段减1，根据IPv6目的地址对报文进行路由查找，最后将报文转发到路由器相应的目的端口。拆封的程序流程如图5。

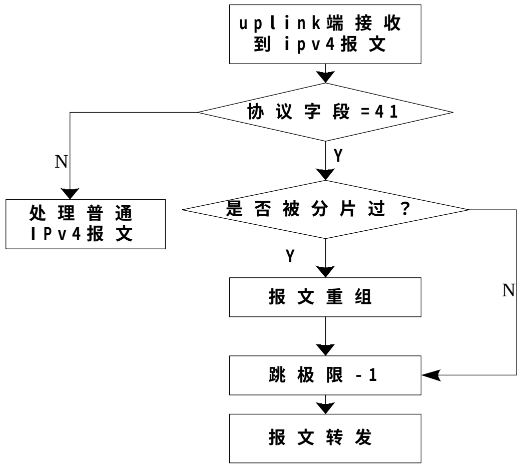

图5 报文拆封流程图

4 结论

本文讨论了IPv4向IPv6转变过程中的一些过渡机制，重点分析了6 to 4 tunnel的一些实现方法问题，阐述了6 to 4 tunnel在IPv6路由器中的实现。IPv6技术及其过渡问题正处于不断研究和探讨阶段，本文只是给出了6 to 4 tunnel的初步实现，关于隧道的安全性以及在实际应用中将会出现的一些问题还有待进一步的补充和完善。

参考文献

1 Gilligan R E. Transition Mechanisms for IPv6 Hosts and Routers. RFC: 2893，2000-08

2 Durand A. IPv6 Tunnel Broker. RFC: 3503，2001-01

3 Carpenter B E. Connection of IPv6 Domains via IPv4 Clouds. RFC: 3056, 2001-02

4 Gilligan R E. Transition Mechanisms for IPv6 Hosts and Routers.RFC: 1933, 1996-04

5 沙 斐. Ipv6详解. 北京: 机械工业出版社，2000

☆☆

（上接第96页）

参考文献

1 Han J, Kambr M. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, 2001: 279-333

2 Ruggieri S. Efficient C4.5. IEEE Transactions on Knowledge and Data Engineering, 2002, 14(2):438-444

3 Breiman L, Friedman JH, Olshen RA, et al. Classification and Regression Trees. Chapman & Hall(Wadsworth, Inc.): New York, 1984

4 Mehta M, Agrawal R, Rissanen J. SLIQ: A Fast Scalable Classifier for Data Mining. Research Report, IBM Almaden Research Center, San Jose, California, 1995

5 Shafer J, Agrawal R, Mehta M. SPRINT: A Scalable Parallel Classifier for Data Mining. Research Report, IBM Almaden Research Center, San Jose, California, 1996

6 Rastogi R, Shim K. PUBLIC: A Decision Tree Classifier that Integrates Building and Pruning. Technical Report, Bell Laboratories, Murray Hill, 1998