文章编号: 1673- 9620 (2008) 04- 0056- 04

C4. 5 算法的两点改进*

乔增伟[1], 孙卫祥[2]
(1. 江苏工业学院 信息科学与工程学院, 江苏 常州 213164; 2. 上海交通大学 振动、冲击、噪声国家重点实验室, 上海 200240)

摘要: C4. 5 作为一种重要的决策树算法尚存一些不足之处。针对 C4. 5 对于连续属性最优分割阈值选择比较耗时的缺点, 基于 Fayyad 边界点判定定理, 提出一种改进最优阈值选择方法。针对 C4. 5 不具备增量式学习能力的缺点, 在改造树结构体的基础上, 提出 C4. 5 增量学习的改进方法。

关键词: 决策树; C4. 5; 最优阈值; 增量式学习
中图分类号: TP 301. 6 文献标识码: A

Two Improvements to C4. 5 Algorithm

QIAO Zeng- wei[1], SUN Wei- xiang[2]
(1. School of Information Science and Engineering, Jiangsu Polytechnic University, Changzhou 213164, China; 2. State Key Laboratory of Vibration, Shock & Noise, Shanghai Jiaotong University, Shanghai 200240, China)

Abstract: As an important decision tree algorithm, C4. 5 still has two disadvantages. One is that it is very time- consuming to find the optimal threshold of continuous attribute. The other is that C4. 5 has no ability of incremental learning. Based on the boundary point theorem given by Fayyad, an improvement method of selecting the optimal threshold is proposed to overcome the first disadvantage. Also, after the modification of tree structure of C4. 5, an incremental C4. 5 algorithm is put forward to solve the second problem.

Key words: decision tree; C4. 5; optimal threshold; incremental learning

决策树的先驱是概念学习系统 (Concept Learning System, CLS), 它由 Hunt 等人提出, 该算法使用一组训练实例, 构造出等价于析取规则形式的决策树表示概念描述, 然后运用这些概念对新的实例进行分类。1986 年, Quinlan 修改了 CLS 并提出的 ID3 算法[1], ID3 以树的形式表示已经获取的知识, 树的内部节点表示测试属性, 叶节点是类标识, 从根到叶形成一条分类规则。1993 年, Quinlan 在文献 [2] 中对 ID3 算法进行了完善, 提出了 C4. 5 算法。

1 C4. 5 算法简介

C4. 5 算法是 ID3 算法的改进, 增加了对连续型属性、属性值空缺情况的处理[3]。算法主体由决策树生成算法 C4. 5tree、剪枝算法 C4. 5pruning、规则生成算法 C4. 5rules 3 部分组成。C4. 5tree 算法依据信息熵理论, 选择当前样本集中具有最大信息增益率的属性作为测试属性不断对样本集进行划

* 收稿日期: 2008- 04- 11
作者简介: 乔增伟 (1978-), 男, 河南驻马店人, 硕士, 讲师, 研究方向: 数据挖掘、软件工程。

分，最终构造出一棵完全决策树。对于连续型的属性先进行离散化，即把连续型属性的值划分成不同的区间，便于处理。C4.5pruning 采用基于错误的剪枝方法对完全决策树进行修剪，得到简化决策树。C4.5rules 把完全决策树转化成一组 if. then 规则集并进行化简。经剪枝或规则生成过程得到的简化决策树和规则集都可用于分类[4]。

C4.5 作为 ID3 的升级版本，虽然已经得到很好的应用，但是还存在一些不足[5]。其一是，对连续属性离散化时，C4.5 算法计算每个分割点的信息熵，从中寻找属性的最优分割阈值，当属性值较多时算法比较耗时；其二是，ID3 经过发展已经出现增量式版本，但是 C4.5 还不具备增量式学习功能。针对这两点不足，对 C4.5 算法进行适当改进，可以提高其性能与适用度。

2 连续属性分割阈值选择方法改进

2.1 C4.5 分割阈值选择方法

连续属性离散化时首先需要找到分割阈值，然后形成离散区间[6]。C4.5 算法进行离散阈值选择的思想是这样的：针对测试属性 X，将其所有属性值按升序或降序排列得 $\{v_1, v_2, \cdots, v_m\}$，分别以 v_i，$i = 2, 3, \cdots, m-1$ 为分段点，将区间 $[v_1, v_m]$ 划分为两个区间 $[v_1, v_i]$ 与 $[v_{i+1}, v_m]$，这样，对于属性 X 存在 $m-1$ 个分割区间，然后检查所有分区的平均类熵（平均类熵的计算方法参见文献[2]），具有最小类熵的阈值视为最优阈值。

从以上离散阈值选择过程可知，当属性 X 的属性值较多时，检查的区间（即计算每个区间类熵）就比较多。

2.2 Fayyad 边界点判定定理

定义 1：属性 A 中的一个值 T 是一边界点，当且仅当在按 A 的值排序的实例序列中，存在两个实例 e_1，$e_2 \in S$ 具有不同的类，使得 $A(e_1) < T < A(e_2)$，且不存在任何其他的实例 $e' \in S$，使得 $A(e_1) < A(e') < A(e_2)$。$A(e)$ 表示实例 e 的 A 属性值。S 表示实例的集合。

定理 1：若 T 使得 $E(A, T, S)$ 最小，则 T 是一个边界点。其中，A 为属性，S 为实例集合，E 表示平均类熵，T 为某一阈值点。定理 1 表明，对连续属性 A，使得实例集合的平均类熵达到最小值的 T，总是处于实例序列中两个相邻异类实例之间。

2.3 改进分割阈值选择方法

由 Fayyad 边界点判定定理可知，无需检查每一个阈值点，只要检查相邻不同类别的边界点即可。为了保持与 C4.5 的一致性，这里边界点选为相邻不同类别的属性值中较小的一个。例如，当排序后的实例属性值为 $\{v_1, v_2, \cdots, v_{10}\}$，其中前 3 个属于类别 C_1，中间 4 个属于类别 C_2，最后 3 个属于类别 C_3，因此只需考察两个边界点 v_3 与 v_7 而无需检查其余 7 个阈值点，然后选择 v_3 与 v_7 中使得平均类熵最小的那个作为最优阈值。

由上可知，改进分割阈值选择方法可以大大减少运算次数，提高训练速度，当需要离散化的属性的属性值越多，而所属类别越少时，性能提高越明显；当出现最不理想情况，即每个属性值对应一个类别，改进算法运算次数与未改进算法相同，不会降低算法性能。

3 C4.5 算法的增量式改进

3.1 增量式学习方式

当前 C4.5 算法存在的主要缺陷就是它不具备增量学习能力，也就是说，当 C4.5 利用某一实例集训练学习生成决策树模型（或导出规则）之后，如果需要将少许新增实例的知识添加到原先模型中，C4.5 是无能为力的，除非利用原有实例及新增实例重新训练学习。这样，对于原先实例数量较大、新增实例数量较少的情况，放弃原有知识、重新学习的代价就比较大。因此，在机器学习方面，增量学习能力非常重要。因为在解决真实世界问题时，很难在训练系统投入使用之前就得到所有可能的训练实例，另外，对一个训练好的系统，进行更新的时间代价通常低于重新训练所需的代价。

增量式学习包括 3 类问题，即 E-IL (Example- Incremental Learning) 问题，C-IL (Class-Incremental Learning) 问题，A-IL (Attribute-Incremental Learning) 问题。

E-IL (Example-Incremental Learning)：学习系统训练好之后，又得到了新的训练样本，需要对训练好的系统进行一定的改动，以使其在尽可能保持已有知识的同时，能够对新知识进行学习。在这个过程中，没有或仅有少量已学习过的训练可

供使用。

C- IL (Class- Incremental Learning): 学习系统训练好之后，输出表示发生了变化，增加了新的输出分类。需要对训练好的系统进行一定的改动，以使其在尽可能保持已有知识的同时，能够对新知识进行学习。在这个过程中，没有或仅有少量已学习过的训练样本可供使用。

A- IL (Attribute- Incremental Learning): 学习系统训练好之后，输入表示发生了变化，增加了新的输出属性。需要对训练好的系统进行一定的改动，以使其在尽可能保持已有知识的同时，能够对新知识进行学习。在这个过程中，没有或仅有少量已学习的训练样本可供使用。

在这 3 种增量式学习方式中，E- IL 是最简单的增量式学习问题，最困难的是 E- IL、C- IL 和 A- IL 同时出现的情况。本文从最简单问题着手，力图解决 C4.5 的 E- IL 增量学习问题。

3.2 C4.5 中树结构的改造

为了实现 C4.5 算法的增量式学习功能，必须对原有树结构作适当的改造。C4.5 的树结构体如下所示:

```
typedef struct _tree_record * Tree;
typedef struct _tree_record
{
  short NodeType; // 节点类型
  ClassNo Leaf; // 叶节点的最优类别
  ItemCount Items, // 节点实例数
    * ClassDist, // 节点实例类别分布情况
    Errors; // 节点分类误差
  Attribute Tested; // 节点测试属性
  short Forks; // 节点的分枝节点数
  float Cut, // 连续属性的最优阈值
    Lower, // 软阈值的下极限
    Upper; // 软阈值的上极限
  Set * Subset; // 离散属性的子集
  Tree * Branch; // 节点的分枝
  }
```

从上述程序可以看出，C4.5 是决策树是单向链表。为了实现增量学习过程中某些节点属性值的更新，需要将单向链表更改为双向链表，于是在结构体中添加一个指向父节点的指针 Father，对于根节点，它的 Father 为空 (NULL)。同时，为了实现增量学习功能，即叶节点知识的更新，需要记忆叶节点的所包含的实例，于是为叶节点添加一个动态数组 SubItems，当分枝节点为叶节点 (NodeType= 0) 时 SubItems 才分配内存空间，其余节点不分配空间。改进后的树结构体如下所示:

```
typedef struct _tree_record * Tree;
typedef struct _tree_record
{
  short NodeType; // 节点类型
  ClassNo Leaf; // 叶节点的最优类别
  ItemCount Items, // 节点实例数
    * ClassDist, // 节点实例类别分布情况
    Errors; // 节点分类误差
  Attribute Tested; // 节点测试属性
  short Forks; // 节点的分枝节点数
  float Cut, // 连续属性的最优阈值
    Lower, // 软阈值的下极限
    Upper; // 软阈值的上极限
  Set * Subset; // 离散属性的子集
  Tree * Branch; // 节点的分枝
  Tree Father; // 添加指向父节点的指针; 根节点的父为 NULL
  Description * SubItems; // 添加记录实例的动态数组，当 NodeType= 0 时分配空间
  }
```

3.3 增量式 C4.5 算法流程

在改造完决策树结构后，就可以考虑 C4.5 的增量学习方法了。当在前一训练模型的基础上需要继续学习即增量学习时，具体的算法步骤如下:

Step1. 输入学习实例;

Step2. 利用已有模型对实例进行分类，经过决策树多层分枝判断，在某一决策节点 DN (DecsionNode 即给出实例所属类别的节点) 上给出决策类别。

If 决策类别与实例类别相同即分类正确，转至 Step3;

Else 分类错误，跳转至 Step5;

Step3. 将学习实例的信息加入到决策节点 DN 即可，包括更新 DN 节点及其父节点和所有上一级父节点的实例数 Items、实例类别分布 ClassDist;

Step4. 在 DN 节点的 SubItems 中添加学习实例，跳转至 Step10;

Step5. 在决策节点 DN 上，利用该节点原有实例及学习实例重新寻找最优测试属性及其最优分割阈

值，最优分割阈值的选择使用改进方法，其他计算过程与 C4. 5 算法相同;

Step6. 在决策节点 DN 上，生成分枝叶节点，Node= Leaf (ClassFreq, BestClass, Cases, Cases- NoBestClass)，并将各个分枝叶节点的父指针 Father 指向 DN，OriginalTree- > Branch [i] = Node; OriginalTree- > Branch [i] - > Father = OriginalTree;

Step7. 设置各个分枝叶节点的节点类别 Leaf、实例数 Items、实例类别分布 ClassDist、节点分类误差 Errors 以及节点包含实例 SubItems;

Step8. 更新原决策节点 DN 及其父节点和所有上一级父节点的实例数 Items、实例类别分布 ClassDist;

Step9. 由于原决策节点 DN 此时不再为叶节点，所以删除该节点所包含的实例 SubItems，free (OriginalTree- > SubItem);

Step10. 学习结束。

4 验证实例及结果分析

4. 1 实验结果

将改进的 C4 5 算法应用到故障诊断中并与 BP 神经网络算法比较，实验使用的分析样本来自转子台模拟故障数据。试验模拟转子 5 种运行状态，即正常、不平衡、径向碰摩，油膜涡动和不平衡+ 径向碰摩。在经过数据采样系统采样后，进行频域幅值谱特征提取。针对每种运行状态各提取 100 个特征样本，于是得到 100 × 5= 500 个样本。实验结果如表 1 所示。

表 1 频域幅值谱特征诊断结果

Table 1 The diagnosis result of frequency features

诊断方法	正常			不平衡			径向碰摩			油膜涡动			径向碰摩+ 不平衡			综合		
	训练	验证	测试	训练	验证	测试	训练	验证	测试	训练	验证	测试	训练	验证	测试	训练	验证	测试
改进 C4 5	100. 0	90 0	89. 0	100 0	89. 5	88 5	100 0	88 5	89. 0	99. 2	87. 0	87. 0	100 0	88 5	88 5	99. 8	88. 2	88 5
BPNN	88 0	77. 0	75. 0	92 5	86 0	92 0	94 0	86 5	93. 5	84 0	80 5	82 0	85. 5	80 0	85. 5	88 8	82. 0	85. 6

4 2 结果分析

由表 1 可以得到如下结论: ① 改进 C4 5 的学习能力较强，BP 神经网络学习能力相对较弱，改进 C4. 5 的训练精度远远高于 BP 神经网络的训练精度。④改进 C4. 5 的泛化能力略高于 BP 神经网络。无论是验证精度还是测试精度，改进 C4 5 均高出 BP 神经网络 2 个百分点以上。

由此不难看出，改进 C4. 5 算法在学习与泛化能力上高于 BP 神经网络，将其引入故障诊断领域，可以提高故障诊断的精度。

5 结 论

本文针对 C4. 5 算法对于连续属性分割阈值选择方法的不足，根据 Fayyad 边界点判别定理，提出了最优阈值选择的改进方法，提高了算法运算速度; 针对 C4. 5 算法不具备增量学习能力的缺陷，在改造树结构体的基础上提出了算法增量学习的改进方法，并将其应用于故障诊断领域，大大提高了 C4. 5 算法的可应用性。

参考文献:

[1] Quinlan J R. Induction of decision trees [J]. Machine Learning, 1986 (1): 84- 100

[2] Quinlan J R. C4 5: Programs for Machine Learning [M]. San Mateo: Morgan Kaufmann Publicshers, Inc, 1993: 17- 42

[3] 冯少荣. 决策树算法的研究与改进 [J]. 厦门大学学报 (自然科学版), 2007, 20 (04): 498- 500

[4] Mehmed Kantardzic. 数据挖掘- 概念、模型、方法和算法 [M]. 北京: 清华大学出版社, 2003: 121- 125

[5] 杨学兵，张俊. 决策树算法及其核心技术 [J]. 计算机技术与发展, 2007, 17 (01): 44- 46

[6] 孙超利. 基于决策树的数据流挖掘算法的研究 [J]. 太原科技大学学报, 2006, 27 (04): 269- 270