文章编号: 0258-2724(2005) 01-0044-05

决策树剪枝方法的比较

魏红宁
(西南交通大学校长办公室, 四川 成都 610031)

摘 要: 为在决策树剪枝中正确选择剪枝方法, 基于理论分析和算例详细地比较了当前主要的 4 种剪枝方法的计算复杂性、剪枝方式、误差估计和理论基础. 与 PEP 相比, MEP 产生的树精度较小且树较大; REP 是最简单的剪枝方法之一, 但需要独立剪枝集; 在同样精度情况下, CCP 比 REP 产生的树小. 如果训练数据集丰富, 可以选择 REP, 如果训练数据集较少且剪枝精度要求较高, 则可以选用 PEP.

关键词: 数据挖掘; 决策树; 事后剪枝; PEP; MEP; REP; CCP
中图分类号: TP311 文献标识码: A

Comparison among Methods of Decision Tree Pruning

WEI Hong-ning
(Administrative Office, Southwest Jiaotong University, Chengdu 610031, China)

Abstract: To select a suitable pruning method in decision tree pruning, four well-known pruning methods were compared in terms of computational complexity, traversal strategy, error estimation and theoretical principle by taking a classification and regression tree as an example. Compared with pessimistic error pruning (PEP), minimum error pruning (MEP) is less accurate and produces a larger tree. Reduced error pruning (REP) is one of the simplest pruning strategies, but it has the disadvantage of requiring a separate data set for pruning. Cost-complexity pruning (CCP) produces a smaller tree than REP with similar accuracy. Practically, if the training data is abundant, REP is preferable; and if the train data is the expected accuracy is high but with limited data, PEP is good choice.

Key words: data mining; decision tree; post pruning; pessimistic error pruning; minimum error pruning; reduced error pruning; cost-complexity pruning

决策树方法是数据挖掘中最为重要的分类方法之一. 决策树是通过对训练数据集重复分组来构造的. 如果训练数据集中的数据能准确地反映分析对象的本质, 则通过该训练数据集所得到的决策树将可以准确地对该问题进行分类. 然而, 由于实际问题中存在许多不确定的因素, 当用决策树构造算法对这类数据分类时, 所得到的决策树将会变得大而复杂, 由此得到的知识规则集也会变得大而复杂. 然而, 研究证明, 大而复杂的决策树并不意味着可以得到更准确的规则集 [1]. 因此, 对决策树进行剪枝非常必要. 当前存在许多种不同的剪枝方法, 分为事前剪枝和事后剪枝两大类 [2], 后者应用得较广泛. 事后剪枝算法又可以分为两类, 一类是把训练数据集分成树生长集和树剪枝集; 另一类算法则在树生长和树剪枝阶段都使用同一训练数据集. 事前剪枝的缺点是使树的生长可能过早停止, 因此应用较少, 所以本文中仅对当前主要的几种事后剪枝算法的应用以及它们的特点和存在的问题进行分析. 讨论中所用的例子(图 1) 是利用 CART (classification and regression trees) 方法 [3] 得到的, 由于训练数据集太大, 在此不列出. 图 1 中 t_i 表示决策树

收稿日期: 2004-03-12
作者简介: 魏红宁(1966-), 男, 高级工程师.

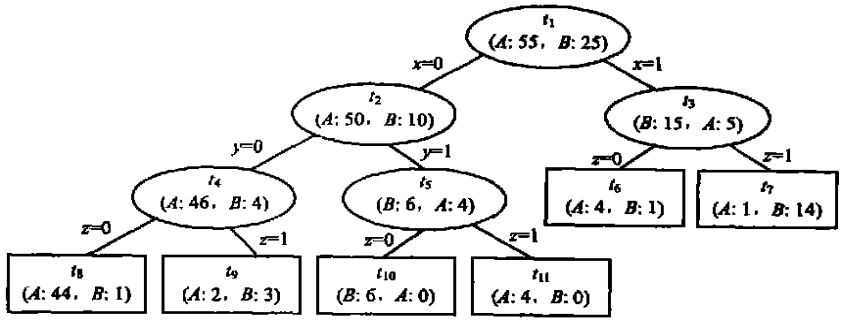


图 1　决策树例子

Fig. 1　Example of decision tree

中第 i 个节点, A, B 表示训练集中的两类, 紧挨的数据为落入该节点分别属于 A, B 类的记录数.

1　CCP(cost-complexity pruning) 方法

CCP 方法[3]主要包含两个步骤: (1) 从原始决策树 T_0 开始生成一个子树序列 T_0, T_1, …, T_n. 其中, T_{i+1}从 T_i 产生, T_n 为根节点. (2) 从第 1 步产生的子树序列中, 根据树的真实误差估计选择最佳决策树.

在步骤 1 中, 生成子树序列$\{T_0, T_1, \ldots, T_n\}$的基本思想是从 T_0 开始, 裁剪 T_i 中关于训练数据集误差增加最小的分枝来得到T_{i+1}. 实际上, 当 1 棵树 T 在节点 t 处剪枝时, 它的误差增加直观上认为是 $R(t)-R(T_t)$, 其中, $R(t)$ 为在节点 t 的子树被裁剪后节点 t 的误差, $R(T_t)$ 为在节点 t 的子树没被裁剪时子树 T_t 的误差. 然而, 剪枝后, T 的叶子数减少了 $|L(T_t)|-1$, 其中, $|L(T_t)|$ 为子树 T_t 的叶子数, 也就是说, T 的复杂性减少了. 因此, 考虑树的复杂性因素, 树分枝被裁剪后误差增加率由下式决定:

$$\alpha = \frac{R(t) - R(T_t)}{|L(T_t)| - 1}. \tag{1}$$

T_{i+1}就是选择 T_i 中具有最小 α 值所对应的剪枝树.

表 1 所示的是关于图 1 的 α 值计算过程及结果.

表 1　决策树的 α 计算值

Tab. 1　Calculated of α on the decision tree in Fig. 1

T_0	$\alpha(t_4)=0.012\,5$	$\alpha(t_5)=0.050\,0$	$\alpha(t_2)=0.029\,2$	$\alpha(t_3)=0.037\,5$
T_1	$\alpha(t_5)=0.050\,0$	$\alpha(t_2)=0.037\,5$	$\alpha(t_3)=0.037\,5$	
T_2	$\alpha(t_3)=0.037\,5$			

从表 1 可以看出, 在原始树 T_0 行, 4 个非叶节点中 t_4 的 α 值最小, 因此, 裁剪 T_0 的 t_4 节点的分枝得到 T_1; 在 T_1 行, 虽然 t_2 和 t_3 的 α 值相同, 但裁剪 t_2 的分枝可以得到更小的决策树, 因此, T_2 是裁剪 T_1 中的 t_2 分枝得到的.

如何从第 1 步产生的子树序列 T_0, T_1, T_2, …中选择出 1 棵最佳决策树是 CCP 方法第 2 步的关键. 通常采用的方法有两种, 一种是 V 番交叉验证(V-fold cross-validation), 另一种是基于独立剪枝数据集. 详见文献[3].

研究表明[4], 生成子树序列 $T(\alpha)$所需要的时间和原决策树非叶节点的关系是二次的, 这就意味着如果非叶节点的数目随着训练例子记录数目线性增加, 则 CCP 方法的运行时间和训练数据记录数的关系也是二次的. 这就比本文中将要介绍的其它剪枝方法所需要的时间长得多, 因为其它剪枝方法的运行时间和非叶节点的关系是线性的.

在 CCP 方法中利用 V 番交叉验证方法, 首先需要将训练数据集 D 划分为 V 个子集 D^1, D^2, …, D^V. 然后, 从 $D-D^i$ ($i=1, 2, \ldots, V$) 中生成 V 个子树 T^1, T^2, …, T^V. $T(\alpha_i)$ 的真实误差就是利用 $T^1(\sqrt{\alpha_i\alpha_{i+1}})$, …, $T^V(\sqrt{\alpha_i\alpha_{i+1}})$的平均值来衡量的. 但这种方法成立的前提是 $T(\alpha_i)$, $T^1(\sqrt{\alpha_i\alpha_{i+1}})$, …, $T^V(\sqrt{\alpha_0\alpha_{i+1}})$具有相同真实误差率. 事实上, 使用自顶向下的决策树算法在小数据集情况下, 认为 T, T^1, …, T^V 具有相同误差率是合理的, 但这种假设不能可靠地扩展到剪枝树 $T^j(\sqrt{\alpha_i\alpha_{i+1}})$中[5].

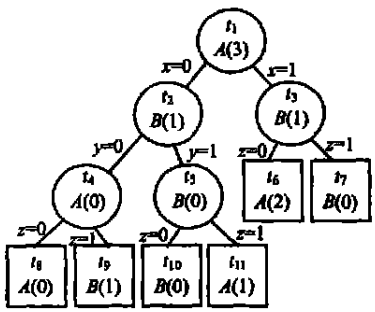
当使用独立剪枝集从 $T(\alpha)$ 中选择最佳决策树时, CCP 方法和 REP 方法相比存在一个缺点, 那就是 CCP 方法只能从 $T(\alpha)$ 中选出最优决策树, 而不是从原始决策树 T 中所有可能的子树中得到. 因此, 如果关于剪枝集的最佳决策树不在 $T(\alpha)$ 中, 则该树不能被得到.

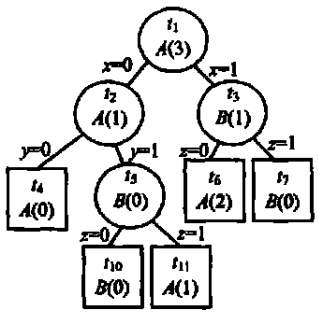
2 REP(reduced error pruning) 方法

REP 方法由 Quinlan 在文献 6]中首先提出, 它需要一个分离数据集 D 用于剪枝. 该方法的基本思路是, 对于决策树 T 的每棵非叶子树S, 用叶子替代这棵子树. 如果 S 被叶子替代后形成的新树关于D 的误差等于或小于 S 关于D 所产生的误差, 则用叶子替代子树 S.

下面以图 1 为例说明 REP 方法如何对决策树进行剪枝的. 图 2(a) 为剪枝数据集, 图 2(b) 和图 2(c) 显示的是基于 REP 方法, 使用图 2(a) 剪枝数据集裁剪图 1 决策树的部分过程.

x	y	z	class
0	0	1	A
0	1	1	B
1	1	0	B
1	0	0	B
1	1	1	A





(a) 剪枝数据集 (b) 剪裁过程 1 (c) 剪裁过程 2

图 2 REP 剪枝示例

Fig. 2 Example of REP algorithm

根据图 1 决策树, 图 2 中的每个节点对剪枝数据集的分类误差在括号中表示. 在遍历树过程中, 采用自底向上的方式, 该方式可以保证剪枝后的结果是关于剪枝数据集的具有最小误差的最小剪枝树.

以图 2(b) 为例, 因为节点 t_4 本身关于剪枝数据集的误差为 0, 而它的子树 t_8 和 t_9 的误差之和为 1. 根据 REP 算法, 则节点 t_4 被转换为叶子, 如图 2(c) 所示. 余下的剪枝过程同上.

REP 是当前最简单的事后剪枝方法之一, 但在数据量较少的情况下很少应用. REP 方法趋于过拟合 (overfitting), 这是因为训练数据集中存在的特性在剪枝过程中都被忽略了, 当剪枝数据集比训练数据集小得多时, 这个问题特别值得注意.

尽管有这些缺点, REP 方法仍常常作为一种基准来评价其它剪枝方法的性能. 它对于了解两阶段决策树学习方法的优点和缺点提供了一个好的初始切入点 [7]. 另外, 它的计算复杂性是线性的, 这是因为决策树中每个非叶节点只需要访问一次就可评估它的子树被剪裁概率. 再者, 由于在剪枝集中的事例没有参与原始决策树的创建, 因此, 和原始决策树相比, 用 REP 剪枝后的决策树对未来事例的预测偏差较少.

3 PEP(pessimistic error pruning) 方法

PEP 方法是 Quinlan [6] 为了克服 REP 方法需要独立剪枝数据集的缺点而提出的, 它不需要分离的剪枝数据集. 为了提高对未来事例的预测可靠性, PEP 方法对误差估计增加了连续性校正(continuity correction). PEP 方法认为, 如果

$$e'(t) \leqslant e'(T_t) + S_e(e'(T_t)) \tag{2}$$

成立, 则 T_t 应被剪裁. 式(2) 中:

$$e'(t) = \left[e(t) + \frac{1}{2}\right]; \tag{3}$$

$$e'(T_t) = \sum e(i) + \frac{N_t}{2}; \tag{4}$$

$$S_e(e'(T_t)) = \left[e'(T_t)\frac{n(t) - e'(T_t)}{n(t)}\right]^{\frac{1}{2}}. \tag{5}$$

其中: $e(t)$ 为节点 t 处误差; i 为覆盖 T_t 的叶子; N_t 为子树 T_t 的叶子数; $n(t)$ 为在节点 t 处训练事例的数.

下面以图 1 为例说明 PEP 在决策树中的剪枝过程, 计算结果如表 2 所示.

表 2　应用 PEP 方法剪裁图 1 的计算结果

Tab. 2　Results on the decision tree in Fig. 1 after PEP

非叶节点	$e'(t)$	$e'(T_t)$	$S_e(e'(T_t))$	是否剪裁
t_1	25.5	8	2.68	否
t_2	10.5	5	2.14	否
t_3	5.5	3	1.60	否
t_4	4.5	4	1.92	是
t_5	4.5	1	0.95	否

PEP 方法采用自顶向下的方式, 如果某个非叶节点的剪裁计算结果符合式(2), 则裁剪该节点的子树. 从表 2 可以看出, 应用 PEP 方法剪裁图 1 决策树的结果是剪裁 t_4 节点的子树.

PEP 方法被认为是当前决策树事后剪枝方法中精度较高的算法之一[2]. 但它仍然存在一些缺陷. 首先, PEP 是唯一使用自顶向下剪枝策略的事后剪枝方法, 这种策略会带来与事前剪枝方法出现的同样问题, 那就是树的某个节点会在该节点的子孙根据同样准则不需要剪裁时也会被剪裁. 另外, PEP 方法有时会剪枝失败[2].

虽然 PEP 方法存在一些局限, 但它在实际应用中表现出了较高的精度. 另外, PEP 方法不需要分离的剪枝数据集, 这对于事例较少的问题非常有利. 再者, 它的自顶向下的剪枝策略使它与其它方法相比效率更高, 速度更快. 这是因为, 在 PEP 方法的剪枝过程中, 树中的每棵子树最多需要访问一次, 在最坏的情况下, 它的计算时间复杂性也只和未剪枝树的非叶节点数目成线性关系.

4　MEP(minimum error pruning)方法

MEP 方法由 Niblett 和 Bratko[8] 首先提出, 该方法使用了拉普拉斯概率估计来提高 ID3 方法在存在噪音数据问题中的性能. Cestnik 和 Bratko[9] 利用贝叶斯方法对这种算法做了一些改进, 称之为 m-概率估计[10]. 在这种算法认为, m 可以根据问题域的不同进行调整. 一般来说, m 越大, 树剪裁程度越深.

归纳起来, MEP 方法的基本思路是采用自底向上的方式, 对于树中每个非叶节点, 首先计算该节点的误差 $E_r(t)$. 然后, 计算该节点每个分枝的误差 $E_r(T_t)$, 并且加权相加, 权为每个分枝拥有的训练样本比例. 如果 $E_r(t)$ 大于 $\sum E_r(T_t)$, 则保留该子树; 否则, 剪裁它.

通常, $E_r(t)$ 的计算采用公式[8]:

$$E_r(t) = \frac{n(t) - n_c(t) + (k-1)}{n(t) + k}, \tag{6}$$

其中: $n(t)$ 为节点 t 中的样本总数; $n_c(t)$ 为 t 中主类的样本数目; k 为类数目.

虽然式(6)是 MEP 初始版本的计算公式, 但该公式简单易懂, 所以通常被采用.

仍以图 1 为例说明 MEP 在决策树中的剪枝过程, 计算结果如表 3.

表 3　应用 PEP 方法剪裁图 1 的计算结果

Tab. 3　Results on the decision tree in Fig. 1 after MEP

非叶节点	$E_r(t)$	$\sum E_r(T_t)$	是否剪裁
T_4	0.096 2	0.081 1	否
T_5	0.416 7	0.141 7	否
T_2	0.17 4	0.146 7	否
T_3	0.272 7	0.159 6	否
T_1	0.317 1	0.117 2	否

MEP 方法不需要独立的剪枝数据集, 无论是初始版本, 还是改进版本, 在剪枝过程中, 使用的信息都来自于训练样本集.

在 MEP 初始版本中, 一个最主要的缺点就是 $E_r(t)$ 和训练样本的类数目 k 相关. 在 MEP 的改进版中, 增加了一项 $(p_{ai}m)$, P_{ai} 是训练样本中第 i 类的先验概率. m 值大小决定了树的剪裁程度. m 值越大, 训练样本集的影响越少, 产生的树越小. 然而, 较高的 m 值并不能自动地生成较小的树, 这种非单调性对 MEP 的计算复杂性影响较大, 也就是说, 对于每一个被考虑的 m 值, 必须从原始树开始剪裁.

显然, 在 MEP 的改进版中, m 的选择非常重要. Cestnik 和 Bratko [9] 建议由领域专家根据不同问题选择 m. 但在大多数情况下, 这是不现实的. 因此, 可以借用 CCP 方法的思路, 利用独立数据集或交叉验证法来选择最佳 m 值. 不过, 这种方式会使 MEP 方法的计算费用显著增加.

5 主要事后剪枝方法的归纳比较

决策树剪枝主要涉及决策树的简化和精度两方面的问题. 决策树剪枝的目的就是在不减少精度的前提下, 简化原始决策树, 从而使获得的知识更加容易理解. 文中第 1 节对当前几种主要的事后剪枝方法的剪枝过程做了重点研究, 归纳起来, 可以通过表 4 中所列举的几方面对它们进行比较.

表 4 事后剪枝方法归纳

Tab. 4 Summary of post-pruning algorithms

比较项目和枝剪方法	CCP	REP	PEP	MEP
独立剪枝集	CV 方式: 不需要	需要	不需要	不需要
剪枝方式	自底向上	自底向上	自顶向下	自底向上
误差估计	使用 CV 或标准误差	利用剪枝集	使用连续性校正	基于 m-概率估计
计算复杂性(n-非叶节点数)	$O(n^2)$	$O(n)$	$O(n)$	$O(n)$

针对图 1 的算例, CCP, REP 和 PEP 给出了剪枝的建议, 但剪枝结果不相同, 而 MEP 方法建议保留图 1 中所有子树. 在实际工作中, 具体选用哪种方法或几种方法的组合, 要根据具体情况. 例如, 如果训练样本集丰富, 则可选用基于独立剪枝集的方法. 本研究主要集中在决策树的剪枝过程, 而决策树剪枝算法的另外一个重要方面就是剪枝之后树的精度问题, 将在以后的工作中作深入研究.

参考文献:

1] Oates T, Jensen D. The effects of training set sizes on decision tree A]. Proc of the 14th Int'l Conf on Machine Learning C]. Nashville: Morgan Kaufman, 1997. 254-262.

2] Breslow L A, Aha D W. Simplifying decision trees: a survey J]. Knowledge Engineering Review, 1997, 12(1): 1-40.

3] Breiman L, Friedman J, Olshen R A, *et al*. Classification and regression trees M]. Belmont: Wadsworth, 1984. 1-358.

4] Nobel A. Analysis of a complexity based pruning scheme for classification trees J]. IEEE Transactions on Information Theory, 2002, 48(8): 2 362-2 368.

5] Malerba D, Semeraro G, Esposito F. Choosing the best pruned decision tree: a matter of bias A]. Proc 5th Italian Workshop on Machine Learning C]. Parma: University of Parma, 1994. 33-37.

6] Quinlan J R. Simplifying decision trees J]. International Journal of Man-Machine Studies, 1987, 27(3): 221-234.

7] Elomaa T, Kaariainen M. An analysis of reduced error pruning J]. Journal of Artificial Intelligence Research, 2001, 15: 163-187.

8] Niblett T, Bratko I. Learning decision rules in noisy domains A]. Proceedings of Expert Systems' 86 C]. Cambridge: Cambridge Uninversity Press, 1986. 25-34.

9] Cestnik B, Bratko I. On estimating probabilities in tree pruning A]. Proc of European Working Sessions on Learning C]. Porto: Springer-Verlag, 1991. 138-150

(中文编辑: 唐 晴 英文编辑: 刘 斌)