基于决策树技术的数据挖掘方法分析和研究

——C4.5 算法的分析和实现

摘要

大数据时代已经到来，对数据的处理越来越受到人们的关注，人们迫切需要海量数据背后的重要信息和知识，发现数据中存在的关系和规则，获取有用的知识，并且根据现有数据对未来的发展做出预测。决策树分类算法 C4.5 算法是数据挖掘中最常用、最经典的分类算法，能够以图形化的形式表现挖掘的结果，从而方便于使用者快速做出决定或预测。决策树实际在各行业应用非常广泛，如客户资源管理（CRM）系统等。本报告从决策树的各个方面对决策树进行分析，理解 C4.5 算法相对于 ID3 算法的改进，并对 C4.5 算法加以实现。同时也指出 C4.5 算法还存在不足。

关键字：数据挖掘；决策树算法；C4.5 算法

一、具体应用场景和意义

决策树（Decision Tree）是用于分类和预测的主要技术，它着眼于从一组无规则的事例推理出决策树表示形式的分类规则，采用自顶向下的递归方式，在决策树的内部节点进行属性值的比较，并根据不同属性判断从该节点向下分支，在决策树的叶节点得到结论。因此，从根节点到叶节点就对应着一条合理规则，整棵树就对应着一组表达式规则。基于决策树算法的一个最大的优点是它在学习过程中不需要使用者了解很多背景知识，只要训练事例能够用属性即结论的方式表达出来，就能使用该算法进行学习。

决策树算法在很多方面都有应用，如决策树算法在医学、制造和生产、金融分析、天文学、遥感影像分类和分子生物学、机器学习和知识发现等领域得到了广泛应用。

决策树技术是一种对海量数据集进行分类的非常有效的方法。通过构造决策树模型，提取有价值的分类规则，帮助决策者做出准确的预测已经应用在很多领域。决策树算法是一种逼近离散函数值的方法。它是一种典型的分类方法，首先对数据进行处理，利用归纳算法生成可读的规则和决策树，然后对新数据进行分析。本质上决策树是通过一系列规则对数据进行分类的过程。

决策树的典型算法有 ID3、C4.5 和 CART 等，基于决策树的分类模型有如下几个特点：（1）决策树方法结构简单，便于理解；（2）决策树模型效率高，对训练集较大的情况较为适合；(3)决策树方法通常不需要接受训练集数据外的知识；（4）决策树方法具有较高的分类精确度。

本报告主要通过分析 C4.5 算法来研究决策树算法。在决策树算法中，最常用的、最经典的是 C4.5 算法，它在决策树算法中的主要优点是：形象直观。该算法通过两个步骤来建立决策树：树的生成阶段和树的剪枝阶段。该算法主要基于信息论中的熵理论。熵在系统学上是表示事物的无序度，是系统混乱程度的统计量。C4.5 基于生成的决策树中节点所含的信息熵最小的原理。它把信息增益率作为属性选择的度量标准，可以得出很容易理解的决策规则。

二、现状分析

决策树技术是迄今为止发展最为成熟的一种概念学习方法。它最早产生于二十世纪 60 年代，是由 Hunt 等人研究人类概念建模时建立的学习系统（CLS，Concept Learning System），到 70 年代末，J Ross Quinlan 提出 ID3 算法，此算法的目的在于减少树的深度。但是忽略了叶子数目的研究。1975 年和 1984 年，分别有人提出 CHAID（Chi-squared Automatic Interaction Detection）和 CART（Classification and Regression Tree，亦称 BFOS）算法。1986 年，J.C.Schlimmer 提出 ID4 算法。1988 年，P.E.Utgoff 提出 ID5R 算法。1993 年，Quinlan 本人以 ID3 算法为基础研究出 C4.5/C5.0 算法，C4.5 算法在 ID3 算法的基础上进行了改进，对于预测变量的缺值处理、剪枝技术、派生规则等方面作了较大的改进，既适合于分类问题，又适合于回归问题。

决策树算法的优点如下：（1）分类精度高；（2）生成的模式简单；（3）对噪声数据有很好的健壮性。因而是目前应用最为广泛的归纳推理算法之一，在数据挖掘中收到研究者的广泛关注。

数据挖掘需要选择复杂度低的算法和并行高效的策略，复杂度低的算法包括尽量把全局最优问题转化成局部最优的问题和近似线性或尽量低阶的多项式复杂度算法等，而高效并行的策略包括需要有高超的递归改为循环的技巧和尽量避免使用全局信息等。

现在研究者们还在继续研究改进的决策树算法，对于 C4.5 算法研究人员们从不同的角度对其进行了相应的改进，其中有针对 C4.5 算法处理连续型属性比较耗时的改进，利用数学上的等价无穷小提高信息增益率的计算效率等等方面。本报告时针对 C4.5 算法本身进行的分析和算法实现，同时会考虑进一步的深入学习。

三、技术挑战分析与解决方案

决策树构造的输入是一组带有类别标记的例子，构造的结果是一棵二叉树或多叉树。二叉树的内部节点(非叶子节点)一般表示为一个逻辑判断，如形式为 a=a_j的逻辑判断，其中 a 是属性，a_j是该属性的所有取值：树的边是逻辑判断的分支结果。多叉树(ID3)的内部结点是属性，边是该属性的所有取值，有几个属性值就有几条边。树的叶子节点都是类别标记。

由于数据表示不当、有噪声或者由于决策树生成时产生重复的子树等原因，都会造成产生的决策树过大。因此，简化决策树是一个不可缺少的环节。寻找一棵最优决策树，主要应解决以下 3 个最优化问题：①生成最少数目的叶子节点；②生成的每个叶子节点的深度最小；③生成的决策树叶子节点最少且每个叶子节点的深度最小。

ID3 算法是一种经典的决策树算法，它从根节点开始，根节点被赋予一个最好的属性。随后对该属性的每个取值都生成相应的分支，在每个分支上又生成新的节点。对于最好的属性的选择标准，ID3 采用基于信息熵定义的信息增益来选择内节点的测试属性，熵（Entropy）刻画了任意样本集的纯度。

ID3 算法存在的缺点：（1）ID3 算法在选择根节点和内部节点中的分支属性时，采用信息增益作为评价标准。信息增益的缺点是倾向于选择取值较多是属性，在有些情况下这类属性可能不会提供太多有价值的信息。（2）ID3 算法只能对描述属性为离散型属性的数据集构造决策树。

ID3 算法的局限是它的属性只能取离散值，为了使决策树能应用与连续属性值，Quinlan 给出了 ID3 的一个扩展算法，即 C4.5 算法。C4.5 算法是 ID3 的改进，其中属性的选择依据同 ID3。它对于实值变量的处理与接下来论述的 CART 算法一致，采用多重分支。C4.5 算法能实现基于规则的剪枝。因为算法生成的

每个叶子都和一条规则相关联，这个规则可以从树的根节点直到叶子节点的路径上以逻辑合取式的形式读出。

决策树的分类过程就是把训练集划分为越来越小的子集的过程。理想的结果是决策树的叶子节点的样本都有同类标记。如果是这样，显然决策树的分支应该停止了，因为所以的类别已经被分开了。

C4.5 算法之所以是最常用的决策树算法，是因为它继承了 ID3 算法的所有优点并对 ID3 算的进行了改进和补充。C4.5 算法采用信息增益率作为选择分支属性的标准，克服了 ID3 算法中信息增益选择属性时偏向选择取值多的属性的不足，并能够完成对连续属性离散化是处理，还能够对不完整数据进行处理。C4.5 算法属于基于信息论（Information Theory）的方法，它是以信息论为基础，以信息熵和信息增益度为衡量标准，从而实现对数据的归纳分类。

C4.5 算法主要做出了以下方面的改进：

（1） 用信息增益率来选择属性

克服了用信息增益来选择属性时偏向选择值多的属性的不足。信息增益率定义为：

$$\text{GainRatio(S, A)} = \frac{Gain(S,A)}{SplitInfo(S,A)} \qquad (1)$$

其中，Grain（S,A）与 ID3 算法中的信息增益相同，而分裂信息 SplitInfo（S, A）代表了按照属性 A 分裂样本集 S 的广度和均匀性。

$$\text{SplitInfo(S, A)} = -\sum_{i=1}^{c} \frac{|Si|}{|S|} Log_2^{\frac{|Si|}{|S|}} \qquad (2)$$

其中，S_1 到 Sc 是 c 个不同值的属性 A 分割 S 而形成的 c 个样本子集。如按照属性 A 把 S 集（含 30 个用例）分成了 10 个用例和 20 个用例两个集合，则 SplitInfo(S,A)=-1/3*log(1/3)-2/3*log(2/3)。

（2） 可以处理连续数值型属性

C4.5 算法既可以处理离散型描述属性，也可以处理连续性描述属性。在选择某节点上的分枝属性时，对于离散型描述属性，C4.5 算法的处理方法与

ID3 相同，按照该属性本身的取值个数进行计算；对于某个连续性描述属性 Ac，假设在某个节点上的数据集的样本数量为 total，C4.5 算法将作以下处理：

- 将该节点上的所有数据样本按照连续型描述的属性的具体数值，由小到大进行排序，得到属性值的取值序列{A1c，A2c，……Atotalc}。
- 在取值序列生成 total-1 个分割点。第 i（0<i<total）个分割点的取值设置为 Vi=（Aic+A（i+1）c）/2，它可以将该节点上的数据集划分为两个子集。
- 从 total-1 个分割点中选择最佳分割点。对于每一个分割点划分数据集的方式，C4.5 算法计算它的信息增益比，并且从中选择信息增益比最大的分割点来划分数据集。

（3） 采用了一种后剪枝方法

避免树的高度无节制的增长，避免过度拟合数据，该方法是用训练样本本身来估计剪枝前后的误差，从而决定是否真正剪枝。方法中使用的公式如下：

$$\Pr\left[\frac{f-q}{\sqrt{q(1-q)/N}} > z\right] = c \qquad (3)$$

其中 N 是实例的数量，f=E/N 为观察到的误差率（其中 E 为 N 个实例中分类错误的个数），q 为真实的误差率，c 为置信度（C4.5 算法的一个熟人参数，默认值为 0.25），z 为对应于置信度 c 的标准差，其值可根据 c 的设定值通过查正态分布表得到。通过该公式即可计算出真实误差率 q 的一个置信区间上限，用此上限为该节点误差率 e 做一个悲观的估计：

$$e = \frac{f+\frac{z^2}{2N}+Z\sqrt{\frac{f}{N}-\frac{f^2}{N}+\frac{z^2}{4N^2}}}{1+\frac{z^2}{N}} \qquad (4)$$

通过判断剪枝前后 e 的大小，从而决定是否需要剪枝。

（4） 对于缺失值的处理

在某些情况下，可供使用的数据可能缺少某些属性的值。假如<x，c(x)>是样本集 S 中的一个训练实例，但是其属性 A 的值 A（x）未知。处理缺少属性值的一种策略是赋给它节点 n 所对应的训练实例中该属性的最常见值；另外一种更复杂的策略是为 A 的每个可能值赋予一个概率。例如，给定一个布尔属性 A，如果结点 n 包含 6 个已知 A=1 和 4 个 A=0 的实例，那么 A(x)=1 的概率是 0.6，而 A(x)=0 的概率是 0.4。于是，实例 x 的 60%被分配到 A=1 的分支，40%被分配到另一个分支。这些片断样例（fractional examples）的目的是计算信息增益，另外，如果有第二个缺失值的属性必须被测试，这些样例可以在后继的树分支中被进一步细分。C4.5 就是使用这种方法处理缺少的属性值

四、系统设计与实现

C4.5 算法的优点是产生的分类规则易于理解，准确率较高。缺点就是在构造树的过程中，需要对数据集进行多次的顺序扫描和排序，因而导致算法的低效。此外，C4.5 算法只适合于能够驻留于内存的数据集，当训练集大得无法在内存容纳时，程序无法运行。

1、C4.5 算法的实现

假设用 S 代表当前样本集，当前候选属性集用 A 表示，则 C4.5 算法 C4.5formtree(S, A)的伪代码如下。

算法：Generate_decision_tree 由给定的训练数据产生一棵决策树

输入：训练样本 samples；候选属性的集合 attributelist

输出：一棵决策树

（1） 创建根节点 N;

（2） IF S 都属于同一类 C，则返回 N 为叶节点，标记为类 C;

（3） IF attributelist 为空 OR S 中所剩的样本数少于某给定值

则返回 N 为叶节点，标记 N 为 S 中出现最多的类;

（4） FOR each attributelist 中的属性

计算信息增益率 information gain ratio;

（5） N 的测试属性 test.attribute = attributelist 具有最高信息增益率的属

性；

（6） IF 测试属性为连续型

则找到该属性的分割阈值；

（7） For each 由节点 N 一个新的叶子节点{

If 该叶子节点对应的样本子集 S’为空

则分裂此叶子节点生成新叶节点，将其标记为 S 中出现最多的类

Else

在该叶子节点上执行 C4.5formtree(S’, S’.attributelist)，继续对它分裂；

}

（8） 计算每个节点的分类错误，进行剪枝。

2、C4.5 算法的基本原理

设 S 是 s 个数据样本的集合。假定类标号 Ci(I = 1,……,m)具有 m 个不同的值，设 s_i 是类 Ci 中的样本数。对一个给定的样本分类所需的期望信息由下式给出：

$$I(s_1,\cdots\cdots,s_m) = \sum_{i=1}^{m} p_i log_2^{(Pi)} \qquad (5)$$

其中，p_i是任意样本属于C_i的概率，并用 s_i/s 来估计。

设属性 A 具有 v 个子集s_1,……, s_v;其中，s_j包含 S 中这样一些样本，它们在 A 上具有值a_j。如果 A 选作测试属性，则这些子集对应于由包含集合 S 的节点生长出来的分枝。设s_{ij}是子集s_j中类C_i的样本数。根据由 A 划分成子集的熵由下式给出：

$$E(A) = \sum_{i=1}^{v} \frac{s_{ij}+\cdots+s_{mj}}{s} I(s_{ij},\ldots\ldots,s_{mj}) \qquad (6)$$

其中，项$\frac{s_{ij}+\cdots+s_{mj}}{s}$充当第 j 个子集的权，并且等于子集（即 A 值为a_j）中的样本个数除以 s 中的样本总数。熵值越小，子集划分的纯度越高。对于给定的子集s_j有：

$$I(S_{1j},S_{2j},\ldots\ldots,S_{mj}) = -\sum_{i=1}^{m} p_{ij} log_2^{Pi_j} \qquad (7)$$

其中，p_{ij}=$\frac{Sij}{Sj}$是s_j中的样本属于类C_i的概率。

在 A 上分枝将获得的编码信息是：

$$Gain(A) = I(s_1,\cdots\cdots,s_m) - E(A) \qquad (8)$$

以上和 ID3 算法的基本原理相同，而 C4.5 所不同的是在后面使用信息增益比例来取代信息增益。

$$SplitInfo(S, A) = -\sum_{i=1}^{c} \frac{|Si|}{|S|} Log_2^{\frac{|Si|}{|S|}} \qquad (9)$$

其中，s_1到s_c是 c 个值的属性 A 分割 S 而形成的 c 个样本子集。

这时，在属性 A 上所得到的信息增益比为：

$$GainRatio(S, A) = \frac{Gain(S,A)}{SplitInfo(S,A)} \qquad (10)$$

C4.5 算法计算每个属性的信息增益比。具有最高信息增益比的属性选作给定集合 S 的测试属性。创建一个节点，并以该属性标记，对属性的每个值创建分枝，并据此划分样本。

3、实例分析

下面我们通过对毕业生就业信息的分析加以理解。在这个分析的结果能够帮助教育者寻找到可能影响毕业生就业的信息，从而在今后的教学过程中进行改进，使得毕业生在就业时更具有竞争力。

学　号	性别	学生干部	综合成绩	毕业论文	就业情况
2000041134	男	是	70~79	优	已
2000041135	女	是	80~89	中	已
2000041201	男	不是	60~69	不及格	未
2000041202	男	是	60~69	良	已
2000041203	男	是	70~79	中	已
2000041204	男	不是	70~79	良	未
2000041205	女	是	60~69	良	已
2000041209	男	是	60~69	良	已
2000041210	女	是	70~79	中	未
2000041211	男	不是	60~69	及格	已
2000041215	男	是	80~89	及格	已
2000041216	男	是	70~79	良	已
2000041223	男	不是	70~79	及格	未
2000041319	男	不是	60~69	及格	已
2000041320	男	是	70~79	良	已
2000041321	男	不是	70~79	良	未
2000041322	男	不是	80~89	良	未
2000041323	女	是	70~79	良	已
2000041324	男	不是	70~79	不及格	未
2000041325	男	不是	70~79	良	未
2000041326	女	是	60~69	优	已
2000041327	男	是	60~69	良	已

表 1　毕业生就业信息

表 1 的数据时经过预处理的数据集，从表中我们可以得到类标号属性“就业情况”有 2 个不同的值（“已”，“未”），因此有 2 个不同的类。其中对应于类值“已”有 14 个样本，类值“未”有 8 个样本。

根据公式（5）我们先计算训练集的全部信息量：

I(就业情况) = I(14, 8) = -14/22log2(14/22)-8/22log2(8/22) = 0.04566030

接着，需要计算每个属性的信息增益比。如以属性“性别”为例：

由公式（5）有：

I(男) = I(10, 7) = -10/17log2(10/17)-7/17log2(7/17)=0.97741728

I(女) = I(4, 1) = -4/5log2(1/5)-1/5log2(1/5)=0.72192809

由公式（6）有：

E(性别) = 17/22*I(男) + 5/22*I(女)=0.91935197

由公式（8）求出这种划分的信息增益：

Gain(性别) = I(就业情况) - E(性别) = 0.02630833

再根据公式（9）求出在该属性上的分裂信息：

SplitInfo(性别) = -17/22log2(17/22)-5/22-log2(55/22)=0.77322667

最后再根据公式（10）求出在该属性上的增益比：

GainRatio(学生干部) = 0.41171446, GainRatio(综合成绩) = 0.08839108, GainRatio(毕业成绩) = 0.10167158

由上述计算结果可知“学生干部”在属性中具有最大的信息增益比，取“学生干部”为根属性，引出一个分枝，样本按此划分。对引出的每一个分枝再用此分类法进行分类，再引出分枝。最后所构造出的判定数如下图所示：

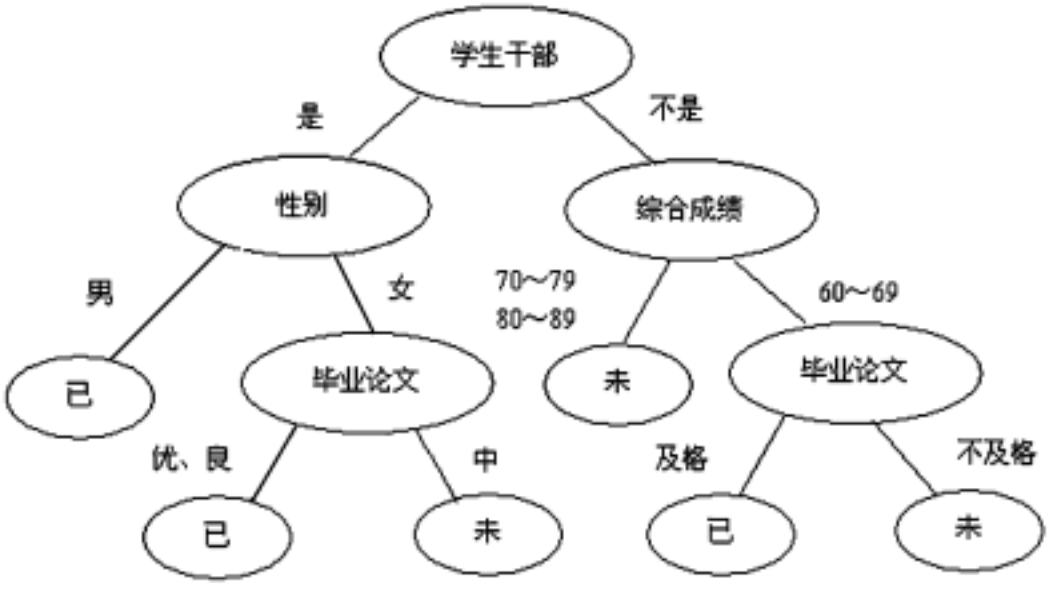

图 1 毕业生就业情况判定树

这是 C4.5 算法的实现，当然 C4.5 算法本身也存在一些不足之处，如处理连续属性比较耗时、计算信息增益率的速度还有待提高等，所以 C4.5 算法还需要不断的研究，并对其加以改进。在以后的学习研究中，我会加深对决策树算法的理解，发现其还存在的不足，提出新的改进方法。

参考文献

[1] Han Jiawei，Micheline K.数据挖掘：概念与技术[M] TP274.范明,孟小峰 译.北京：机械工业出版社,2001:70-218

[2] 毛国君，段立娟，王实.数据挖掘原来与算法[M].北京：清华大学出版社,2005

[3] Quinlan J R. C4.5:Programs for Machine Learning[M]. NewYork:Morgan Kaufnan,1993

[4] Quinlan J R. Induction of decision tree[J]. Machine Learning 1986, 1(1):81-106

[5] 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），2007，17（5）：16-18

[6] 李慧慧,万武族. 决策树分类算法 C4.5 中连续属性过程处理的改进[M] TP301. 1006-2475(2010)08-0008-03

[7] 黄爱辉. 决策树 C4.5 算法的改进及应用[J].科学技术与工程,2009,9(1):34-36

[8] J.R.Quinlan. Improved Use of Continuous Attributes in C4.5[J]. Journal of Artificial Intelligence Rearch 4 (1996) 77-90