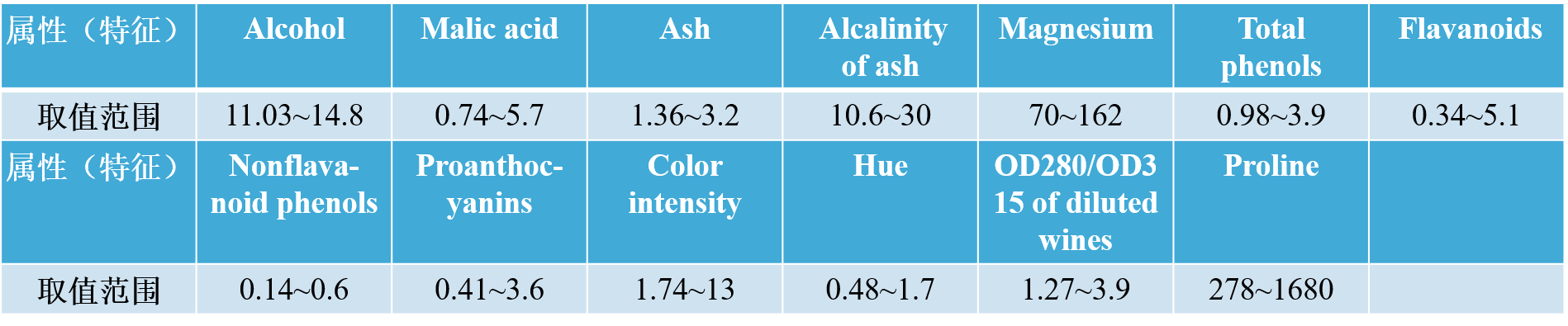
**决策树算法对UCI wine数据集分类的实现**

# 1 功能实现与样本分析

在数据挖掘领域，可以利用相应的算法对数据集进行训练，即对样本的特征进行分析，从而归纳出相同类别的样本之间存在的内在特征联系，进一步对未知类别的样本进行预测，判断出该样本所属的类别。本文实现了利用决策树算法对UCI 机器学习库上的经典wine数据集进行分类的目的。

为达到相应的分类目的，需要先对数据集样本进行分析。

表1 wine数据集样本特征



不用化学成分的含量与酒的所属类别息息相关。表1列举了所选用的wine数据集的特征情况。整个数据集样本包含13个特征，例如Alcohol(酒精度)，Malic acid（苹果酸含量）等，这些特征的取值范围见表1所示。

表2 wine数据集部分样本表

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 属  性  1 | 属性  2 | 属性  3 | 属性  4 | 属性  5 | 属性  6 | 属性  7 | 属性  8 | 属性  9 | 属性  10 | 属性  11 | 属性  12 | 属性  13 | 类别 |
| 14.23 | 1.71 | 2.43 | 15.6 | 127 | 2.8 | 3.06 | .28 | 2.29 | 5.64 | 1.04 | 3.92 | 1065 | 1 |
| 13.2 | 1.78 | 2.14 | 11.2 | 100 | 2.65 | 2.76 | .26 | 1.28 | 4.38 | 1.05 | 3.4 | 1050 | 1 |
| 13.16 | 2.36 | 2.67 | 18.6 | 101 | 2.8 | 3.24 | .3 | 2.81 | 5.68 | 1.03 | 3.17 | 1185 | 1 |
| 14.37 | 1.95 | 2.5 | 16.8 | 113 | 3.85 | 3.49 | .24 | 2.18 | 7.8 | .86 | 3.45 | 1480 | 1 |
| 13.24 | 2.59 | 2.87 | 21 | 118 | 2.8 | 2.69 | .39 | 1.82 | 4.32 | 1.04 | 2.93 | 735 | 1 |
| 14.2 | 1.76 | 2.45 | 15.2 | 112 | 3.27 | 3.39 | .34 | 1.97 | 6.75 | 1.05 | 2.85 | 1450 | 1 |
| 14.39 | 1.87 | 2.45 | 14.6 | 96 | 2.5 | 2.52 | .3 | 1.98 | 5.25 | 1.02 | 3.58 | 1290 | 1 |
| 12.37 | .94 | 1.36 | 10.6 | 88 | 1.98 | .57 | .28 | .42 | 1.95 | 1.05 | 1.82 | 520 | 2 |
| 12.33 | 1.1 | 2.28 | 16 | 101 | 2.05 | 1.09 | .63 | .41 | 3.27 | 1.25 | 1.67 | 680 | 2 |
| 12.64 | 1.36 | 2.02 | 16.8 | 100 | 2.02 | 1.41 | .53 | .62 | 5.75 | .98 | 1.59 | 450 | 2 |
| 13.67 | 1.25 | 1.92 | 18 | 94 | 2.1 | 1.79 | .32 | .73 | 3.8 | 1.23 | 2.46 | 630 | 2 |
| 12.37 | 1.13 | 2.16 | 19 | 87 | 3.5 | 3.1 | .19 | 1.87 | 4.45 | 1.22 | 2.87 | 420 | 2 |
| 12.17 | 1.45 | 2.53 | 19 | 104 | 1.89 | 1.75 | .45 | 1.03 | 2.95 | 1.45 | 2.23 | 355 | 2 |
| 12.37 | 1.21 | 2.56 | 18.1 | 98 | 2.42 | 2.65 | .37 | 2.08 | 4.6 | 1.19 | 2.3 | 678 | 2 |
| 12.86 | 1.35 | 2.32 | 18 | 122 | 1.51 | 1.25 | .21 | .94 | 4.1 | .76 | 1.29 | 630 | 3 |
| 12.88 | 2.99 | 2.4 | 20 | 104 | 1.3 | 1.22 | .24 | .83 | 5.4 | .74 | 1.42 | 530 | 3 |
| 12.81 | 2.31 | 2.4 | 24 | 98 | 1.15 | 1.09 | .27 | .83 | 5.7 | .66 | 1.36 | 560 | 3 |
| 12.7 | 3.55 | 2.36 | 21.5 | 106 | 1.7 | 1.2 | .17 | .84 | 5 | .78 | 1.29 | 600 | 3 |
| 12.51 | 1.24 | 2.25 | 17.5 | 85 | 2 | .58 | .6 | 1.25 | 5.45 | .75 | 1.51 | 650 | 3 |

酒的品种可能属于3种类别。表2展示了wine数据集的部分样本情况。所以，本文的目的就是运用分类算法，挑选相应的特征属性来构造相应的决策树，挖掘wine数据集类别与13个特征之间的关系。并利用所构造的决策树来对测试样本做预测，分析预测误差，从而证明所构造的决策树的合理性。

# 2 C4.5算法对UCI wine数据集分类的实现

## 2.1 C4.5算法分类流程设计



图1 C4.5算法构造决策树流程设计

C4.5算法是决策树分类领域的一种较为经典的算法，图1展示了本次使用C4.5算法构造决策树的算法流程。

首先，从UCI 机器学习库上下载wine数据集，提取样本。这里将所有样本分为训练样本和测试样本两部分。训练样本用来训练C4.5算法，测试样本用来最后测试C4.5算法所构造的决策树的正确性。所以样本预处理部分便是将训练样本与测试样本分开。本文采用的是随机采样的方法来分开训练样本和测试样本。

然后，将随机采样所得到的训练样本用来构造决策树。依次对训练样本的各个特征计算信息增益率，选择具有最大信息增益率的特征作为当前样本的分割特征，对决策树进行分支。

如果决策树分支之后，所有的分割样本都满足预先定义的决策树剪枝策略，则完成对训练样本构造决策树的目标。否则将利用分割样本递归构造决策树，直到所有的分割样本都满足剪枝策略为止。



图2 测试样本类别预测流程图

C4.5算法构造决策树后，需要对决策树的合理性做评估。图2展示了利用所构造的决策树预测测试样本类别流程。通过对决策树的预测误差作分析，来判别所构造的决策树是否可行。下面将介绍具体的决策树构造过程。

## 2.2 C4.5算法构造决策树

### [2.2.1 树节点设计](#直接计算生成函数法测定稳定常数的原理)



图3 决策树节点设计

在决策树的构造过程中，信息增益率的计算显得至关重要。各个特征的信息增益率的大小决定了当前节点的分割特征。因此，在决策树的节点设计上考虑了这一点。图3显示了决策树节点的所有成员属性。Feature\_tosplit用来记录当前节点的分割特征，location记录分割特征的分割界限，value表征的是属于当前节点的特征的取值情况，而child描述按照feature\_tosplit和location进行分支后，隶属于当前节点的子节点。需要注意的是，如果当前节点已经是叶子节点，则利用child属性来记录当前节点的标签，相应地，将feature\_tosplit设置为0，将location和value设置为空。

### [2.2.2 特征取值连续处理](#直接计算生成函数法测定稳定常数的原理)

当属性类型为离散型，无须对数据进行离散化处理；而当属性类型为连续型时，则需要对数据进行离散化处理。为此，节点分裂之前先要对属性的类型做判断。

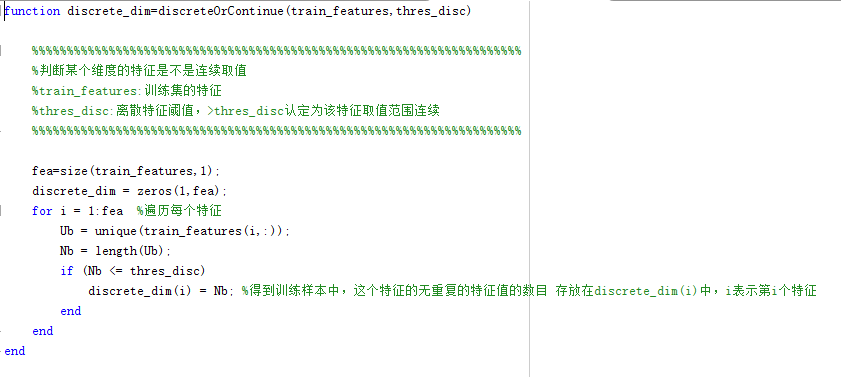


图4 属性类型判断

图4是对某一维度属性类型判断的相关代码。Thres\_disc是为了判断属性是离散类型还是连续类型而设立的一个阈值，在算法的运行过程中，设定thres\_disc=10。当某一维度属性的所有可能取值情况多于thres\_disc时，认为该属性的类型是离散型；否则认为是连续型。

对于连续型的属性，本文利用C4.5算法离散化的核心思想是：将属性A（假设属性A是连续取值）的N个属性值按升序排列；通过二分法将属性A的所有属性值分成两部分（共有N-1种划分方法，二分的阈值为相邻两个属性值的中间值）；计算每种划分方法对应的信息增益，选取信息增益最大的划分方法的阈值作为属性A二分的阈值。

详细流程如下：

1. 将节点Node上的所有数据样本按照连续型属性A的具体取值，由小到大进行排列，得到属性A的属性值取值序列；
2. 在序列中共有N-1种二分方法，即共产生N-1个分隔阈值。对于第*i*种二分方法，其二分阈值**。它将该节点上的数据集划分为2个子数据集和。计算此种二分结果下的信息增益。
3. 分别计算N-1种二分结果下的信息增益率，选取信息增益率最大的二分结果作为对属性A的划分结果，并记录此时的二分阈值。



图5 连续属性离散化处理及相应信息增益率的计算

图5是对连续属性离散化处理的相关代码。对N-1个可能的分割点分别计算信息增益率将耗费很多计算时间，大大提高了算法的时间复杂度。对于这一点，后文的4.3小节还会做进一步的讨论。

### [2.2.3 剪枝策略](#直接计算生成函数法测定稳定常数的原理)

由于决策树的建立完全是依赖于训练样本，因此决策树对训练样本能够产生完美的拟合效果。但这样的决策树对于测试样本来说过于庞大而复杂，可能产生较高的分类错误率。这种现象就称为过拟合。因此需要将复杂的决策树进行简化，即去掉一些节点解决过拟合问题。

本文在防止决策树出现过拟合的问题上采用的是预剪枝的方法：定义一个阈值(pruning)，当达到某个节点的实例个数小于阈值时就可以停止决策树的生长。

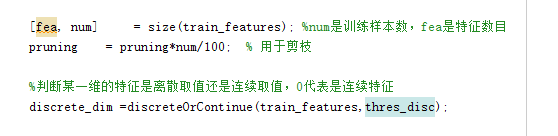


图6 剪枝阈值的设定

对于剪枝阈值(pruning)的设定，使其与训练样本的个数成正比，而不是设定成一个固定的值，这样可以方便地处理各种容量的样本数据。

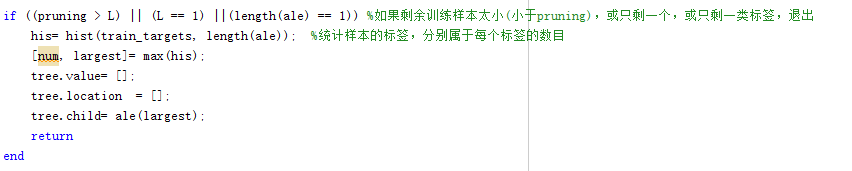


图7 剪枝阈值用于控制决策树的构造

最后将剪枝阈值用于控制决策树的生长：当剩余的训练样本数太少时，不让当前节点继续分裂，并将剩余训练样本所属类别最多的类别作为当前节点的最终标签。

### [2.2.4 决策树构造结果](#直接计算生成函数法测定稳定常数的原理)

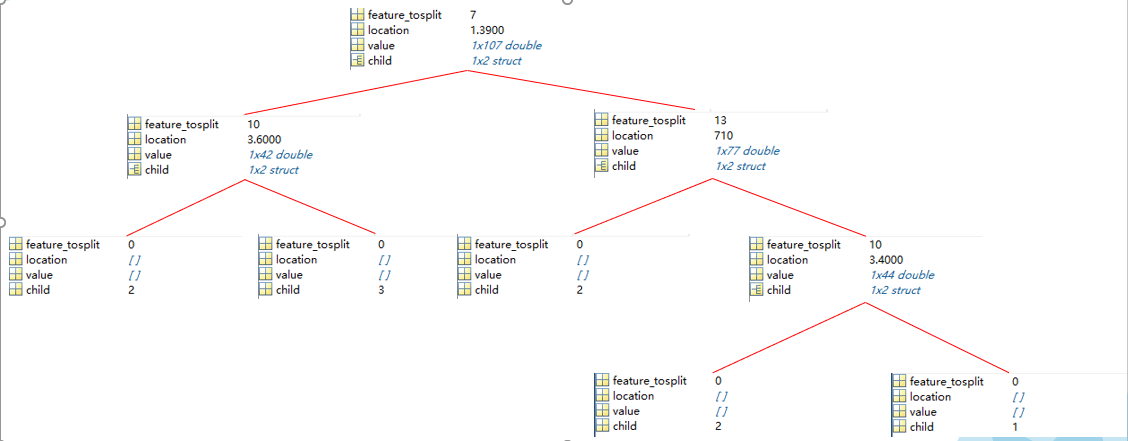


图8 C4.5算法所构造决策树结构及节点属性

按照前文所描述的C4.5算法构造决策树的相关操作，并利用MATLAB编程实现对wine数据集的决策树构造，最终得到如图8所示的决策树结构。各个节点的属性所代表的含义如2.2.1小节所述。



图9 C4.5算法所构造决策树

将图8所示结构转化为图9更为明了的决策树示意图。下面将对所构造的决策树的合理性进行评估。

## 2.3 C4.5算法的分类准确度



图10 C4.5算法分类准确度分析

把随机采样之后剩余的部分样本作为测试样本，利用图9所示的决策树对测试样本做预测，来评估C4.5算法构造决策树的分类准确度。

图10是用MATLAB运行100次的实验结果。在这100次的测试中，基本上决策树的预测准确度均在80%以上，预测的准确度均值约为92%，已经能够提取大部分的有效特征来对wine做分类，实现了基本的分类需求。

# 3 C4.5算法与CART算法的比较

本文着重实现了C4.5决策树算法对于wine训练集的分类，也在第2节中用了大量篇幅讲解了具体的实现细节。但是提到决策树算法，又不得不提及CART算法。故本节将对比C4.5算法和CART算法在wine数据集的训练效果，但对于CART算法的具体细节将不再过多阐述。



图11 CART算法所构造决策树



图12 C4.5和CART算法的分类准确度比较

图11呈现了利用CART算法最终所构造的决策树。与图9 C4.5算法的决策树结构存在差异。但两者在分割特征的选取上有很大的相似性，比如都充分利用了属性7，属性10以及属性13来构造决策树。

图12是50次实验下，利用图9和图11对测试数据的分类准确度比较。需要注意的是，每次实验，两种算法运用的训练集和测试集虽然都是随机采样的得到的，但是两者的数据是完全相同的。否则对于两种算法的准确度比较将没有意义。

总体而言，CART决策树的分类准确度会高于C4.5所构造的决策树。原因可能在于本文采样的预剪枝的剪枝策略有待改进，剪枝阈值的设定可能不是最优从而导致C4.5决策树过早地停止生长。如果以后要进一步地优化本文的C4.5决策树分类策略，可以从这点出发对剪枝策略进行改进。

4 实验分析与讨论

## 4.1 准确度问题

首先分析的是，某些测试条件下，分类准确度异常的现象。

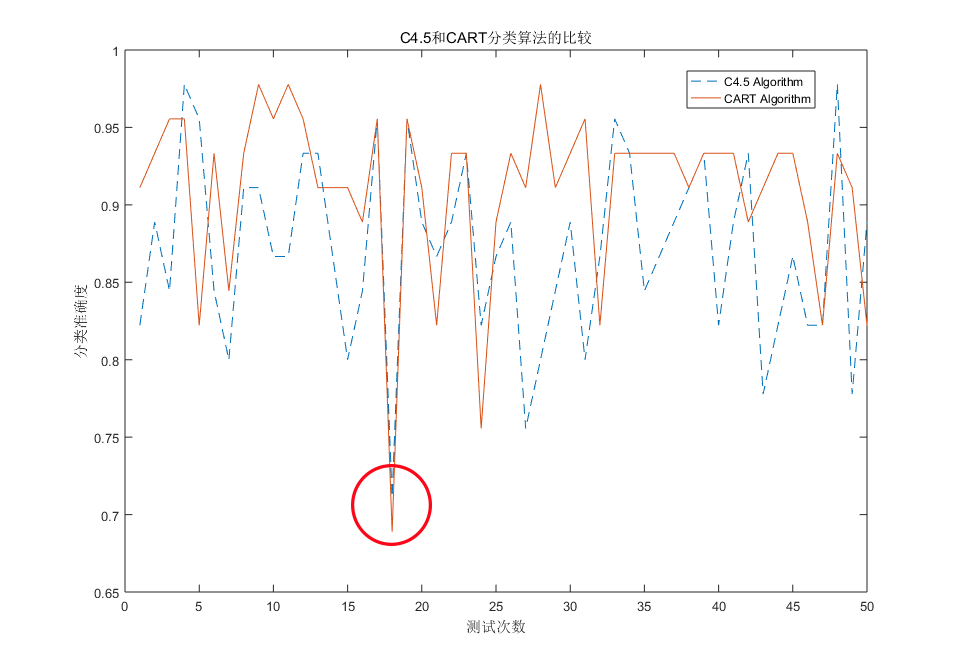


图13 准确度异常现象

再次考虑图12两种分类算法的准确度对比结果。可以发现，虽然大多数情况下两种算法的预测准确度可以达到85%甚至更高，但是也存在着某些点准确度很低的情况。正如图13红色圆圈所圈出来的点。

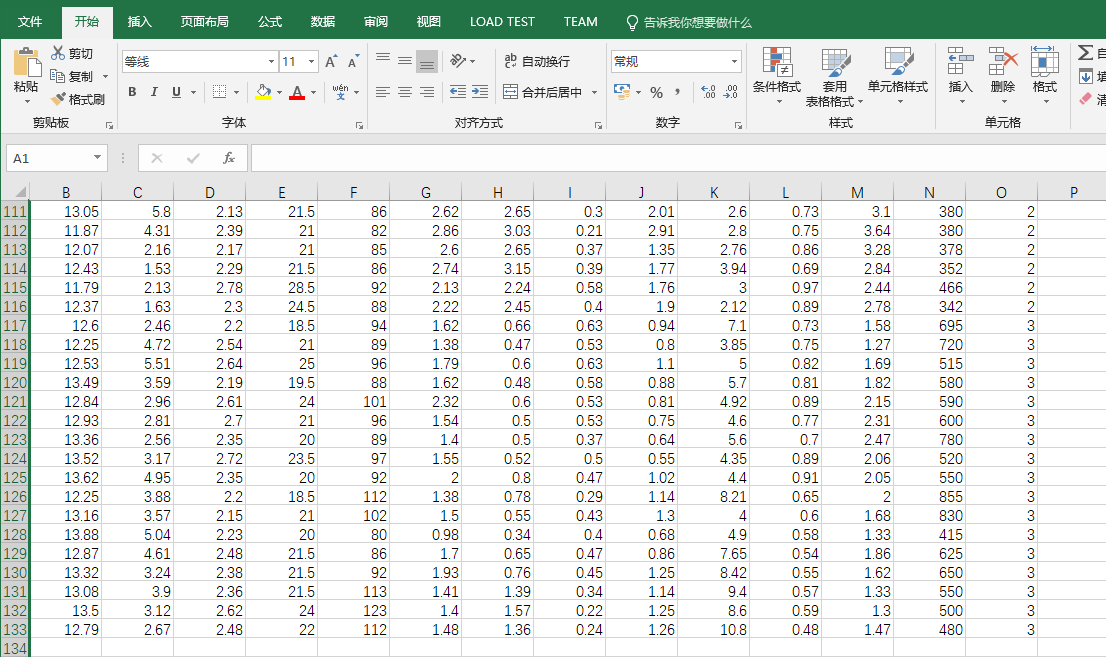


图14 采样不均匀样本

图15 样本采样不均匀分析

最终发现问题出在随机采样。本文之所以采用随机采样的方法，是因为这样训练出来的决策树更具有普适性：它可以减少人工分离训练样本和测试样本所带来的训练误差。

但是随机采样也有可能带来新的问题。正如图14和图15所示，采样训练样本的时候可能存在采样不均匀的现象：某些类别的训练样本数有可能很低。这样一来，就很难找到相应的特征对这一类别的样本做分类，随之就有可能出现图13所示某些情况决策树分类准确度较低的现象。

随机采样的优越性很明显，如果每次训练都固定训练样本，则有可能训练出来的决策树只适合当前样本而不具备实用性。但同时也要考虑随机采样带来的新问题。一个解决办法就是多进行几次训练，训练之后准确度分析取平均或者剔除异常数据，减少采样不均匀带来的影响。

## 4.2 缺省值问题

在本文利用C4.5算法对UCI wine数据集分类过程中，并未出现特征缺少特征值的现象。但是现实中的数据却往往可能因为保存不周等原因造成数据集某些特征值的缺失。特征参数的缺失不仅影响模型的训练，还会影响我们利用模型进行样本预测。本小节对缺省值问题进行讨论。

对于缺省值处理的问题，主要需要解决的是两个问题：一是在样本某些特征缺失的情况下选择划分的属性；二是选定了划分属性，对于在该属性上缺失特征的样本的处理。

对于第一个子问题，假设现有缺失特征值的特征A。解决思路可以将数据分成两部分，对每个样本设置一个权重（初始值设为1），然后划分数据，一部分是有特征值A的数据D1，另一部分是没有特征A的数据D2。然后对于没有缺失特征A的数据集D1计算A特征的信息增益率，最后乘上一个系数，等于特征A无缺失的样本占总样本的比例。

对于第二个子问题，可以将缺失特征的样本同时划分入所有的子节点，不过将该样本的权重按各个子节点样本的数量比例来分配。比如缺失特征A的样本a之前权重为1，特征A有3个特征值A1，A2，A3。3个特征值对应的无缺失A特征的样本个数为2，3，4。则a同时划分入A1，A2，A3，对应权重调节为2/9，3/9，4/9。接着递归构造决策树直到决策树构造完成。

## 4.3 算法复杂性问题

在C4.5构造决策树的过程中，节点的分割属性的选择至关重要。C4.5算法使用的是信息增益率作为特征的分割标准。然而信息增益率的计算较为复杂，这一点在对连续属性的处理上被进一步地凸现出来。



图16 算法运行时间分析

图16展示了C4.5算法和CART算法在处理本文选择的wine数据集的程序运行时间的对比。其中，程序的运行时间可以通过调用MATLAB的tic和toc做差得到。图16的小图对应的是CART算法运行时间的局部放大图。

从中可以明显看出，C4.5算法的运行时间远远大于CART算法的执行时间，这一差距随着样本容量的变大将被进一步的拉大。

为了更为高效地建立决策树用于分类，可以从运用简化的二叉树模型替代多叉树模型和用近似的基尼系数来取代信息增益率来简化计算两方面着手。减少算法的执行时间。

# 参考文献

[1] 李航. 统计学习方法[M]. 北京：清华大学出版社，2012.

[2] 郭炜星. 数据挖掘分类算法研究[D]. 浙江大学，2008.

[3] Jiawei Han, Micheline Kamber.Data Mining Concepts and Techniques[M]. Morgan Kaufmann

Publishers Inc. 2011:70-218.

[4] 刘庆和，梁正友. 一种基于信息增益的特征优化选择方法[J]. 计算机工程与应用，2011，47

（12）：130-132.

[5] 杨志辉. 基于机器学习算法在数据分类中的应用研究[D]. 太原：中北大学，2017.

[6] 张亮，宁芊. CART决策树的两种改进及应用[J]. 计算机工程与设计. 2015，05.

[7] Chen Jin, Luo De-Lin, Mu Fen-xiang. An improves ID3 decision tree algotithm.IEEE Computer

Science and Education, ICCSE’09.4th International Conference. 2009, 127-130.

[8] 冯帆, 徐俊刚. C4.5决策树改进算法研究[J]. 电子技术研究，2007:1-4.