

模式识别

课程报告

姓 名：

班 号：

学 号：

院 系： 计算机学院

专 业：计算机科学与技术

任课教师： 蒋良孝

2019 年 5 月

目录

[一、决策树概述 3](#_Toc8146075)

[1.1决策树和ID3算法 3](#_Toc8146076)

[1.2 C4.5算法 3](#_Toc8146077)

[二、C4.5基本思想 3](#_Toc8146078)

[2.1生成决策树 3](#_Toc8146079)

[2.2特征选择 4](#_Toc8146080)

[2.2.1信息增益 4](#_Toc8146081)

[2.2.2增益率 4](#_Toc8146082)

[2.3剪枝优化 5](#_Toc8146083)

[2.3.1先剪枝 5](#_Toc8146084)

[2.3.2后剪枝 5](#_Toc8146085)

[2.4 连续值和缺省值处理 6](#_Toc8146086)

[2.4.1连续值处理 6](#_Toc8146087)

[2.4.2缺省值处理 6](#_Toc8146088)

[三、运行结果与分析 7](#_Toc8146089)

# 一、决策树概述

## 1.1决策树和ID3算法

决策树是一种基于树形结构，对样本做出决策而进行分类的一种机器学习算法。在决策树上，每个结点都保存了部分或者全部样本，每次对某个属性做出决策而进行分类都会将父节点分裂成若干子节点。然后子节点作为新的父亲节点继续分裂，直至不能分裂为止。每次分裂都是而根节点是没有父节点的节点，即初始分裂节点，叶子节点是没有子节点的节点。

决策树的基本算法有ID3，ID3算法需要解决的问题是如何选择特征作为划分数据集的标准。在ID3算法中，选择信息增益最大的属性作为当前的特征对数据集分类。通过不断的选择特征对数据集划分；结束情况分为三种，第一种是当划分出来的样本属于同一个类；第二种为已经没有属性可供再分了；第三种是当前样本为空，即所有样本已经分完了，此时也结束了。

## 1.2 C4.5算法

C4.5算法是用于生成决策树的一种经典算法，是ID3算法的一种延伸和优化。C4.5算法对ID3算法主要做了一下几点改进：（1）通过信息增益率来选择分裂属性，克服了ID3算法中通过信息增益倾向于选择拥有多个属性值的属性作为分裂属性的不足；（2）构造决策树之后进行剪枝操作；（3）能够处理离散型和连续型的属性类型，即将连续型的属性进行离散化处理；（4）能够处理具有缺失属性值的训练数据。

# 二、C4.5基本思想

## 2.1生成决策树

决策树生成算法的核心是在每个决策树各个节点上应用相应的算法选出最佳特征，根据该特征的不同取值，递归的构造决策树。如果该节点样本为空，或者所有样本全部属于同一分类结果则停止递归。除此之外，如果达到某个设定的停止条件，那么也应该停止递归。

C4.5通过增益率来选择最佳特征。具体算法如下

定义生成函数generate(D, A)将节点D用属性A划分，生成子节点

（1）若当前节点所有样本属于同一类，将节点标记为该类，返回

（2）若当前节点为空或所有属性已划分，则将节点记为样本数最多的分类结果，返回

（3）否则从未使用的属性中选出最佳属性，对于该属性的每个取值，都生成分支。根据样本的取值，将其划分到相应的结点。

（4）如果某个子节点没有样本，则将其标记为父节点样本数最多的分类结果。

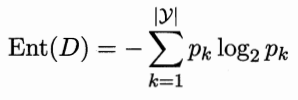
（5）否则将当前最佳属性标记为已使用，转（1）

## 2.2特征选择

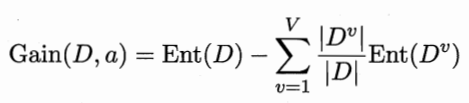
决策树算法一个重要步骤是在样本中选出最好的那个特征，然后对样本进行区分。我们的目标是每次生成的子节点所包含的样本尽量属于同一类别，也就是纯度越来越高。选择的算法有很多，ID3是基于信息增益，分类回归树是基于基尼指数，而C4.5是基于增益率。

### 2.2.1信息增益

上面我们谈到划分的最好属性是尽可能将分到的该属性不同取值的样例都属于同一类，要使得其纯度最高，我们可以考虑用信息熵来衡量。假设样本集合D中第k个样本所占比例为pk，则该样本集合的信息熵可以定义为如下公式



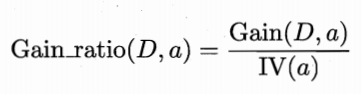
信息熵反映的是混乱程度，如果集合的信息熵越大则其混乱程度越大，反之如果Ent(D)的值越小，则表明其纯度越高。评估每个属性值的好坏，我们需要看该属性下分出来的子节点的集合的混乱程度，这里我们采用的公式如下，其中V是属性a的不同取值。对于属性a的每个取值赋予一个权重，然后求和。样本数量占比越多的，其信息熵也应该占比越大。最后将原始父节点的样本集合的信息熵减去该值，即为信息增益。



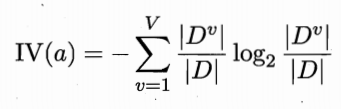
### 2.2.2增益率

使用信息增益会遇到一个问题，如果我们不小心将样本的序号也作为一个属性进行训练了，那么很显然只需要这一个属性即可将所有的样本都分开，但这种决策树是没有意义的。如果一个属性其取值很多，那么基于信息增益的决策树会更倾向于选择该属性，因为属性取值越多，那么越容易分出纯净的子节点。但是这样却降低了模型的泛化性能，用增益率可以解决这一问题。

增益率在信息增益的基础上增加了一个惩罚值，该属性取值越多，则该惩罚越大。具体公式如下Gain\_ratio是样本集合在属性a上的增益率。



IV(a)就是该属性的惩罚值，公式如下，其中V为属性a的取值，Dv为属于v这一属性的样本集合。



从公式中我们可以看出，如果一个属性其取值很多，那么|Dv|/|D|就越小，则IV(a)就越大，即惩罚越大。但是增益率的问题是其更倾向于属性取值少的属性，所以我们在选择属性的时候先找信息增益高于平均水平的，在从中选择增益率最高的。

## 2.3剪枝优化

决策树创建过程中，由于数据中的噪声和离群点，许多分支反应的是训练数据中的异常。剪枝方法是用来处理这种过分拟合的问题，通常剪枝方法都是使用统计度量，减去最不可靠的分支。剪枝方法有先减枝和后剪枝。

### 2.3.1先剪枝

先剪枝是在生成决策树的时候进行的。首先我们假定每个结点的分类结果为当前节点样本集合中属于该分类结果的样本数最多的那个。比如一个样本集合中有10个样本，其中7个是分类结果A，另外3个是分类结果B，则我们该节点的分类结果为A。那么如果不继续划分生成子节点，这个结点的分类正确率就是70%。

然后我们选出最优属性，对其进行划分，生成子节点。接着计算出所有的子节点的分类结果，并得到所有子节点中分类正确的样本。我们再次计算分类正确率——分类正确样本的个数占父节点的样本个数的比例。如果生成子节点了以后如果分类的正确率下降了，我们就不进行分类，即进行剪枝。

除了上述剪枝方法以外，常见的先剪枝方法还有如下几种。

（1）当决策树达到一定的高度就停止决策树的生长。

（2）到达节点的样本个数小于某值的时候停止树的生长

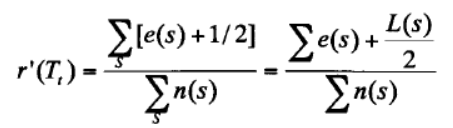
（3）计算每次扩展对系统性能的增益，如果小于某个阈值就可以停止树的生长。

### 2.3.2后剪枝

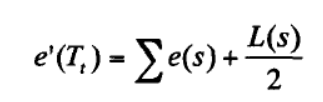
先剪枝虽然可以较好的解决过拟合的现象，但是该剪枝利用的是贪心的思想。当前会使分类正确率下降不一定总体的正确率就会下降。可能在接下来的划分中反而使得正确率显著提高。而且先剪枝还可能导致很多节点无法展开，使模型泛化性降低，发生欠拟合的现象。

为了解决上述问题，我们可以使用后剪枝。后剪枝的具体思路是先生成决策树，然后对生成的决策树进行剪枝，通过对比剪枝前后分类错误率来验证是否进行剪枝。C4.5算法采用的剪枝算法被称为悲观剪枝法。

我们设某一叶子节点t样本个数为n(t)，其中分类错误的个数为e(t)。S为树T的子树Tt，其叶子节点的个数为L(S)，则Tt的分类误差公式可以表示为如下



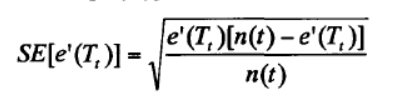
为了方便计算，我们用错误总数代替错误率



如果满足如下条件，则发生剪枝，应该用叶子节点替换父节点



SE[e’(Tt)]被称为标准误差



## 2.4 连续值和缺省值处理

### 2.4.1连续值处理

在学习任务中，常常会遇到连续的属性。由于连续属性取值是无穷的，我们不能像离散属性那样生成节点。此时，我们需要将连续值离散化，也就是设置若干阈值。只要该属性值都在相邻两阈值之间，可视其为同一属性值。通过这种方法，可以将连续值离散化。C4.5算法利用的二分法对连续属性进行处理，具体过程如下。

假设在训练数据中，该属性有n个取值，分别为。我们取n-1个点为阈值点，其中第i个点的取值为。这种处理方式将连续取值分成了n个区间。属性取值在区间(的样本都可视为同一属性值。

### 2.4.2缺省值处理

在现实任务中，我们会遇到不完整的样本，即样本的某些属性值是缺失的。如果直接舍弃有缺省值的样本，会极大的浪费数据信息。一般缺失样本在训练中会存在如下的三个问题。

1. 对于缺失某属性值的样本，如何计算该属性的増益率？

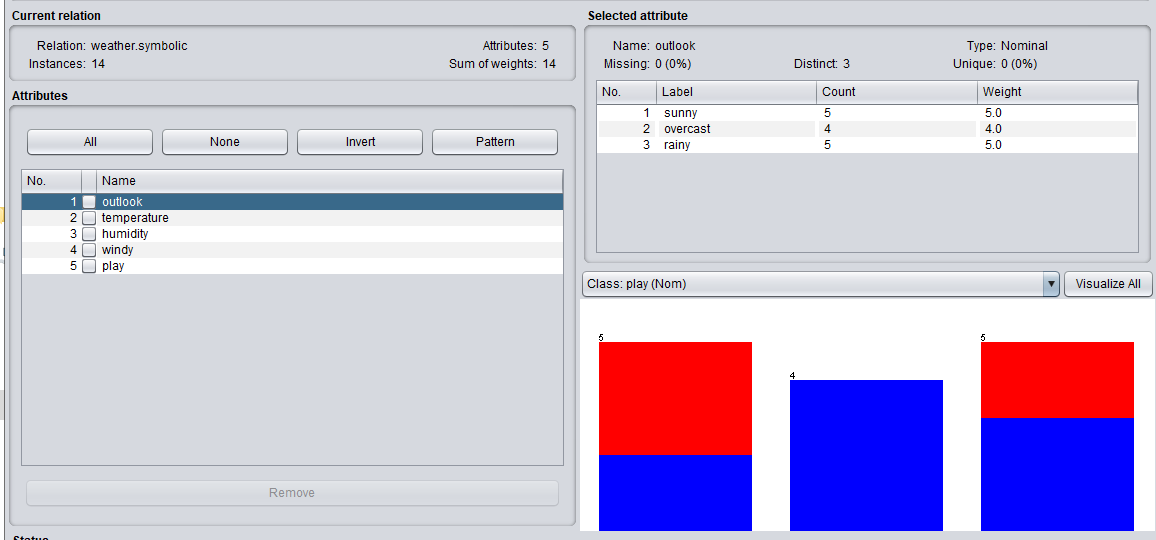
一般有如下几种方式：忽略该样本，即对于该属性而言不参与增益率的计算；将样本中此属性的出现频率最高的属性值赋值给该样本；将同类样本中此属性的出现频率最高的属性值赋值给该样本；为缺省值属性的每个可能值赋予一个概率。

2. 已确定划分属性，对于该属性的值缺失的样本，应该如何划分？

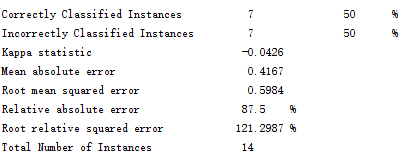
不处理那些属性Ａ未知的样本，即简单的忽略它们；或者根据属性Ａ的其他样本的取值，来对未知样本进行赋值；或者为缺失属性Ａ的样本单独创建一个分支，不过这种方式得到的决策树模型结点数显然要増加，使模型更加复杂了。

# 三、运行结果与分析

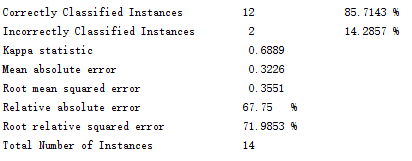
测试使用的数据集是weka中提供的weather数据集



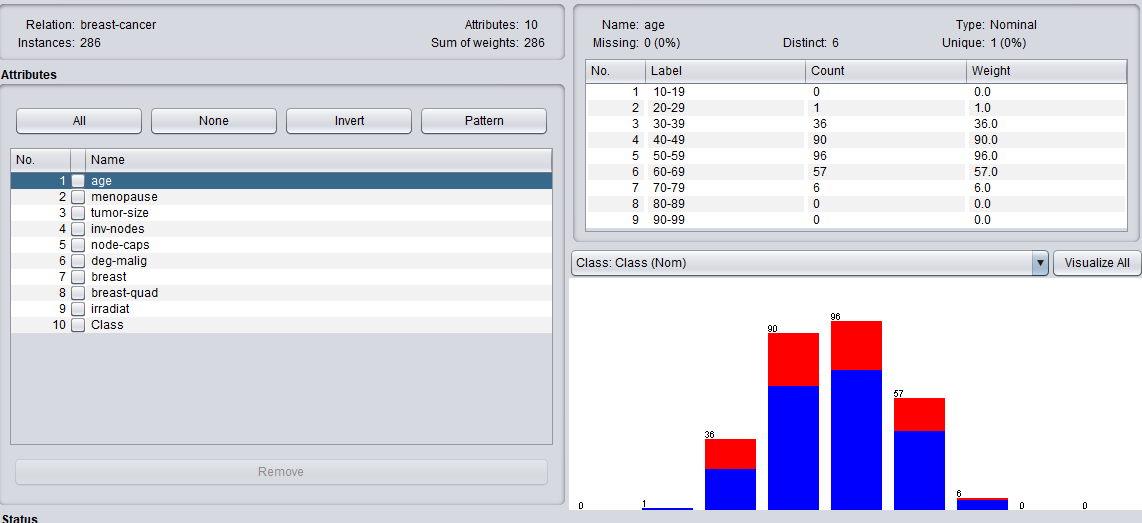
这个是weka中的J48



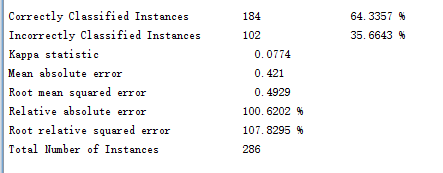
这个是我实现的C4.5



由于上面你个数据集比较小，用于测试的数据也比较少，下面重新测了另一组较大的数据



这个是我实现的C4.5



这个是weka中的J48

