算法与复杂性报告 3：决策树

516030910259 刘欣鹏

1. 决策树模型简介

决策树可视作一个 If...then 规则的集合，是一种类似流程图的树结构。其中每个内部非叶节点表示在一个属性上的测试，每一个分支代表一个测试输出，每个叶子节点代表一个最终的类标号。

建立好决策树后，对于一个需要被分类的输入，从根开始向下进行一条从根节点到叶节点的路径，最终的叶节点即存放该输入的预测分类。

决策树的优点是计算复杂度不高，输出结果便于理解，对中间值的缺失不敏感，可以处理不相关的特征数据，缺点是可能会产生过匹配问题。

2. 构造算法

本次报告中采取 ID3 算法进行决策树的构建。思路如下：

1) 创建树的根节点
2) 若样例分类标签一致，则返回该标签的叶节点
3) 求出属性中分类能力最佳（信息增益最大）的属性 A
4) 对于属性 A 的每一个可能值 v：
    a. 在根节点下增加分支 A=v 的节点
    b. Example(i)为样本集合中 A=v 的子集
    c. 递归调用 ID3(Example(i),属性集-{A})

3. 主要代码

```
def split(dataSet,axis,value)://提取 dataSet 中第 axis 个属性为 value 的样本的子集并返回
def chooseBest(dataSet)://计算分割后可获最大信息增益的属性并返回
def createTree(dataSet,labels):
    Classes=[example[-1] for example in dataSet]//提取当前样本集分类标签
    if Classes.count(Classes[0])==len(Classes):
        return Classes[0]//若标签相同，则返回叶节点
    if len(dataSet[0])==1:
        return majorCnt(Classes)//若无属性可用于分割，则返回叶节点，取标签中最多的一个做为叶节点标签
    best=chooseBest(dataSet)
    bestlabel=labels[best]
    //计算最适合分割的属性 A
    myTree={bestlabel:{}}
    del(labels[best])//删除属性 A
    Values=[x[best] for x in dataSet]
    Vals=set(Values)//提取 A 的所有可能值
    for value in Vals:
        subLabels=labels[:]
        myTree[bestlabel][value]=createTree(split(dataSet,best,value),subLabels)
        //按 A 的所有可能值分割样本集并递归建树
    return myTree
def classify(Tree,labels,test)://对测试数据预测分类
    x=Tree.keys()[0]
    y=Tree[x]
```

```
i=labels.index(x)
for key in y.keys():
    if test[i]==key:
        if type(y[key]).__name__=='dict':
            ans=classify(y[key],labels,test)
            //当前节点非叶则继续跟踪路径
        else:
            ans=y[key]
            //当前节点为叶子则返回标签
return ans
```

4. 实例应用

4.1. 数据集

	age	prescipt	astigmatic	tear rate	conclusion
1	young	myope	no	reduced	no lenses
2	young	myope	no	normal	soft
3	young	myope	yes	reduced	no lenses
4	young	myope	yes	normal	hard
5	young	hyper	no	reduced	no lenses
6	young	hyper	no	normal	soft
7	young	hyper	yes	reduced	no lenses
8	young	hyper	yes	normal	hard
9	pre	myope	no	reduced	no lenses
10	pre	myope	no	normal	soft
11	pre	myope	yes	reduced	no lenses
12	pre	myope	yes	normal	hard
13	pre	hyper	no	reduced	no lenses
14	pre	hyper	no	normal	soft
15	pre	hyper	yes	reduced	no lenses
16	pre	hyper	yes	normal	no lenses
17	presbyopic	myope	no	reduced	no lenses
18	presbyopic	myope	no	normal	no lenses
19	presbyopic	myope	yes	reduced	no lenses
20	presbyopic	myope	yes	normal	hard
21	presbyopic	hyper	no	reduced	no lenses
22	presbyopic	hyper	no	normal	soft
23	presbyopic	hyper	yes	reduced	no lenses
24	presbyopic	hyper	yes	normal	no lenses

其中，age、prescript、astigmatic、tear rate 为四个属性，conclusion 为分类标签。本报告中，以 1、11、23 为测试集，其他为训练集。

4.2. 训练与测试

4.2.1. 训练结果

```
{'tear_rate': {'reduced': 'no', 'normal': {'astigmatic': {'yes': {'prescript': {
'hyper': {'age': {'pre': 'no', 'presbyopic': 'no', 'young': 'hard'}}, 'myope': '
hard'}}, 'no': {'age': {'pre': 'soft', 'presbyopic': {'prescript': {'hyper': 'so
ft', 'myope': 'no'}}, 'young': 'soft'}}}}}
>>>
```

上图中为以 python 字典形式存储的决策树。

4.2.2. 测试结果

```
>>> tree.classify(t,labels,['young','myope','no','reduced'])
'no'
>>> tree.classify(t,labels,['pre','myope','yes','reduced'])
'no'
>>> tree.classify(t,labels,['presbyopic','hyper','yes','normal'])
'no'
```

与表格对比，发现与之基本一致。