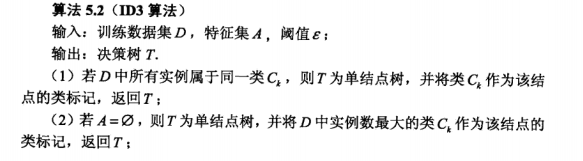
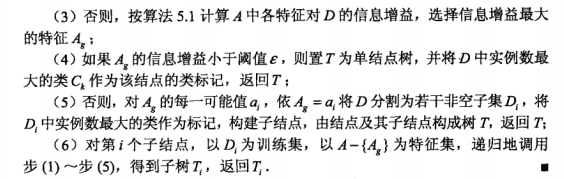
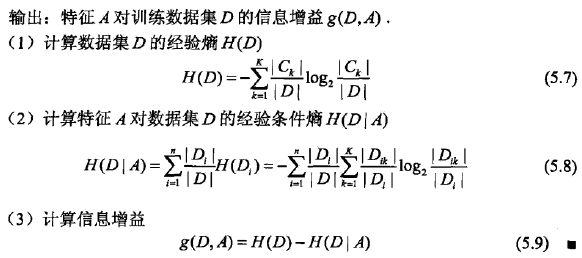
【算法】



信息增益



在数据比较大，特征比较多的情况下，很容易造成过拟合，于是需进行决策树剪枝，一般剪枝方法是当按某一特征分类后的熵小于设定值时，停止分类。

**【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 。

为了改进决策树，又提出了ID4.5算法（依据：信息增益比）和CART算法（回归树：平方误差最小/分类树：基尼指数最小）。

【案例：头发声音判断特征】

参考博客

（案例）http://blog.csdn.net/csqazwsxedc/article/details/65697652

（代码注解详细） <http://blog.csdn.net/u011197534/article/details/52749325?hmsr=toutiao.io&utm_medium=toutiao.io&utm_source=toutiao.io>

（另个一类似案例）<http://blog.csdn.net/alvine008/article/details/37760639>

（其他）http://blog.csdn.net/wds2006sdo/article/details/52849400

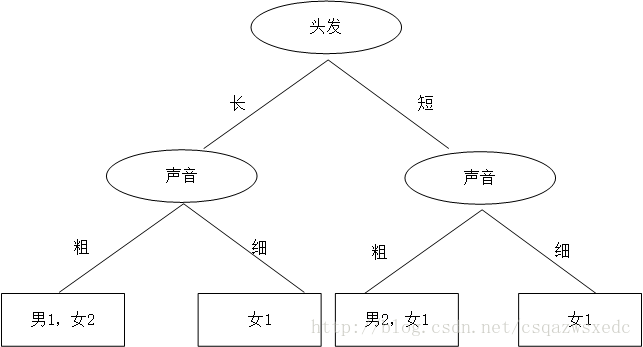
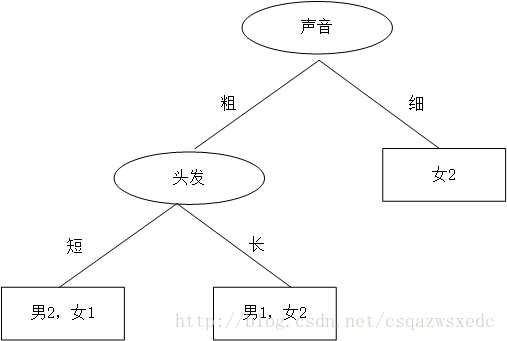
A、问题描述

一天，老师问了个问题，只根据头发和声音怎么判断一位同学的性别。

为了解决这个问题，同学们马上简单的统计了7位同学的相关特征，数据如下：

|  |  |  |
| --- | --- | --- |
| 头发 | 声音 | 性别 |
| 长 | 粗 | 男 |
| 短 | 粗 | 男 |
| 短 | 粗 | 男 |
| 长 | 细 | 女 |
| 短 | 细 | 女 |
| 短 | 粗 | 女 |
| 长 | 粗 | 女 |
| 长 | 粗 | 女 |

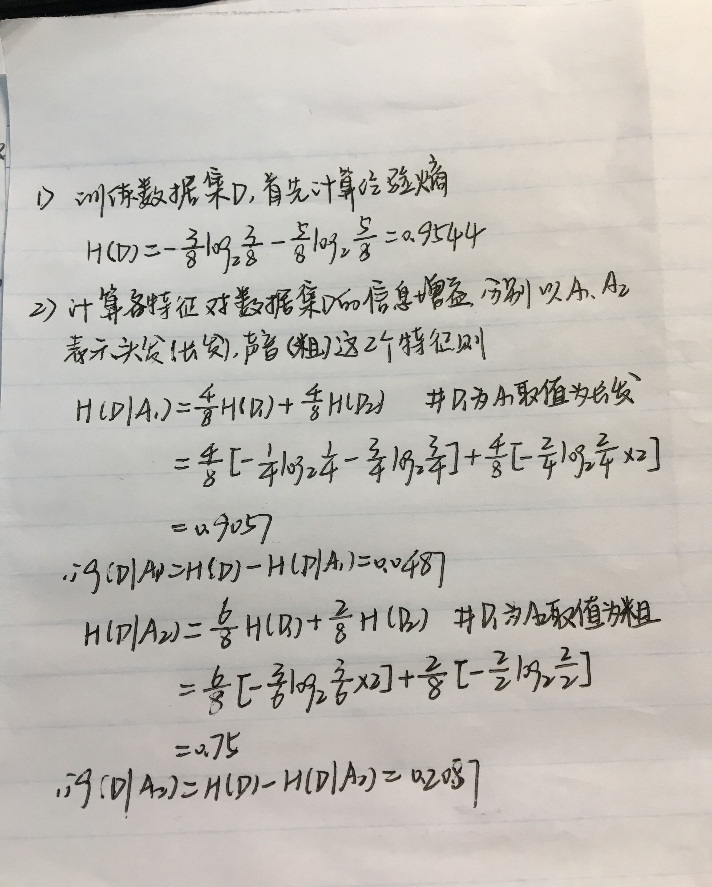
B、计算解决

同学A 同学B

信息增益的计算

总共有8位同学，男生3位，女生5位。



比较各特征的信息增益值。由于A2（声音）的信息增益值更大，所以选择A2为最优特征。

C、代码

1.创建示例数据

def createTrainset():

dataSet = [['长', '粗', '男'],

['短', '粗', '男'],

['短', '粗', '男'],

['长', '细', '女'],

['短', '细', '女'],

['短', '粗', '女'],

['长', '粗', '女'],

['长', '粗', '女']]

train\_set = [sample[:-1] for sample in dataSet]

train\_label = [sample[-1] for sample in dataSet] #['男', '男', '男', '女', '女', '女', '女', '女']

features = ['头发','声音']

return np.array(train\_set),np.array(train\_label),features

2.预处理：统计label的的个数，放入字典class\_count{label:count}。

def createclass\_count(train\_label):

class\_count = {}

for label in train\_label:

if label in class\_count:

class\_count[label] += 1

else:

class\_count[label] = 1

return class\_count

3.计算原始信息熵和各特征对数据集的信息熵

1）计算信息熵hd

def calc\_ent(train\_label):

hd = 0

class\_count = createclass\_count(train\_label)

for label in class\_count.keys():

prob = class\_count[label] / len(train\_label)

hd -= prob \* math.log(prob,2)

return hd

2）计算各特征对数据集的信息熵had = H(train\_label | feature)

#其中输入feature为train\_set的一列e.g ['长', '短', '短', '长', '短', '短', '长', '长']

def calc\_condition\_ent(feature,train\_label):

hda = 0

for feature\_value in set(feature):

#对于某个特征feature\_value,只考虑包含此特征的train\_label

sub\_train\_label = train\_label[feature == feature\_value]

tempt\_ent = calc\_ent(sub\_train\_label)

hda += len(sub\_train\_label) / len(train\_label) \* tempt\_ent

return hda

4.递归构建决策树

def createTree(train\_set,train\_label,features):

# 步骤1——如果train\_set中的所有实例都属于同一类Ck[递归停止条件]

label\_set = set(train\_label)

if len(label\_set) == 1:

return label\_set.pop()

# 步骤2——如果特征集features为空[递归停止条件]

class\_count = createclass\_count(train\_label)

# → 类标记为train\_set中实例数最大的类

max\_class = max(class\_count, key=class\_count.get)

if len(features) == 0:

return max\_class

# 步骤3——计算信息增益

#计算原始的信息熵

hd = calc\_ent(train\_label)

max\_gda = 0

max\_feature\_idx = -1

#计算各特征对数据集D的信息增益

for i in range(len(features)):

gda = hd - calc\_condition\_ent(train\_set[:,i],train\_label)

if gda > max\_gda:

max\_gda = gda

max\_feature\_idx = i

# 步骤4——小于阈值

if max\_gda < epsilon:

return max\_class

# 步骤5——构建非空子集

max\_feature = features[max\_feature\_idx]

tree = {max\_feature:{}}

#删除此特征

del(features[max\_feature\_idx])

feature\_value = set(train\_set[:,max\_feature\_idx]) #构建子树,遍历feature的值

for value in feature\_value:

#子树的训练集和label只包含-含有当前feature-的行

sub\_train\_set = train\_set[train\_set[:,max\_feature\_idx] == value]

sub\_train\_label = train\_label[train\_set[:,max\_feature\_idx] == value]

tree[max\_feature][value] = createTree(sub\_train\_set,sub\_train\_label,features)

return tree

\*判定分类结束的依据是，若按某特征分类后出现了最终类（男或女），则判定分类结束。

5.测试\_输入特征，判断性别

def classify(myTree,features,sample):

#找到最优特征-根节点['声音']

firstfeature =list(myTree.keys())[0]

#找到最优特征的索引

feature\_idx = features.index(firstfeature) #此处'声音'的索引为1

#判断最优特征的值

secondDic = myTree[firstfeature]

for key in secondDic.keys():

if key == sample[feature\_idx]: #和sample对应处相同

if type(secondDic[key]) == dict:#还没有结束

mylabel = classify(secondDic[key],features,sample)

else:

mylabel = secondDic[key]

return mylabel

6.main函数及输出

if \_\_name\_\_ == '\_\_main\_\_':

epsilon = 1e-6

train\_set,train\_label,features = createTrainset()

myTree = createTree(train\_set,train\_label,features)

print(myTree)

train\_set,train\_label,features = createTrainset()

sample = ['短','粗']

mylabel = classify(myTree,features,sample)

print('特征为%s的同学性别为: %s'%(str(sample),mylabel)) # 男

**输出：**

{'声音': {'细': '女', '粗': {'头发': {'长': '女', '短': '男'}}}}

特征为['短', '粗']的同学性别为: 男