中山大学数据科学与计算机学院

移动信息工程专业-人工智能

本科生实验报告

（2017-2018 学年秋季学期）

课程名称：**Artificial Intelligence**

教学班级	**M1**	专业（方向）	**移动信息工程（互联网方向）**
学号	**15352102**	姓名	**韩硕轩**

一、 实验题目

决策树（decision_tree）

二、 实验内容

1. 算法原理

用一组有 n 个特征和 1 个标签的训练向量，ID3 或 C4.5 或 CART 的决策方法，递归构建一棵决策树。对于每一个树上的节点，有一个子数据集和子特征集，以上三种决策方法可以选出一列最有区分度的特征，然后将数据集根据特征划分，并向下递归。这样相当于写了很多个 if-else 语句[1]，只不过我们不自己写，而是教会代码去构建。

当做验证集和测试集的时候，就从根节点开始，选择对应向量的对应特征，向下匹配，直到叶节点，返回对应的类别即为我们用决策树方法预测的答案。

2. 伪代码

（0）Ent：（算熵）

```
Ent(t, ...) = -( t / total ) * log( t / total) - …
```

（1）ID3：

```
Id3(data_set, remain):
    For I in data_set:
        统计 1 的个数 t
        统计-1 的个数 f
    经验熵 H(D) = Ent(t,f)
    For p in columns:
        If p 被用过: continue
        For I in data_set:
            统计特征取值数 a
        For I in data_set:
            统计每个特征取值向量数 b
            统计每个特征取值正确向量数 c
        计算条件熵 H(D|A)=b/向量数*Ent((b-c)/b, c/b)
    选择最大的增益列返回
```

（2）C4.5：

```
C45(data_set, remain):
    //信息增益部分省略，以下是增益率部分
    For p in columns:
        For I in 特征向量数 b:
            Splitinfo += Ent(b[i])
        增益率 ratio = 增益 / splitinfo
    选择增益率最大的特征列返回
```

（3）CART：

```
CART(data_set, remain):
    For p in columns:
        Data_set 向量数 l
        For I in data_set:
            统计特征取值数 a
        For I in data_set:
            统计每个特征取值向量数 b
            统计每个特征取值正确向量数 c
        For I in a:
            Gini += b/l*(1-(c/b)^2-((b-c)/b)^2)
    选择最小的基尼系数列返回
```

（4）decision_tree：

```
Decision_tree(tree, data_set, remain):
    If data_set 类别都是 1 或-1:
        返回 1 或-1
    If remain 为空:
        返回 data_set 中最多的类别
    If data_set 在某一特征上取值相同:
        返回 data_set 中最多的类别
    从 remain 中选择一个特征 p
    Remain[p] = 0
    For I in data_set:
        统计特征取值数 a
    For I in a:
        给每个特征取值添加对应的向量到 b
    For I in b:
        Tree(空树, b[i], remain)
```

（5）calc：

```
Calc(tree, data):
    获取当前节点对应的特征 x
    If 是叶结点:
        返回 data[x]
    Else:
        Calc(data[x], data)
```

（6）main：

读取训练集 ts

划分验证集 vs

初始化 remain 为 1

初始化决策树 root = {}

Decision_tree(root, ts, remain)

For I in vs：

    Calc(root, i)

    统计正确率

读取测试集 cs

For I in cs:

    输出 calc(root, i)

3. 关键代码截图（带注释）

按照伪代码顺序展示关键代码截图。

首先是 id3 关键部分：

```
hd = 0
t = 0
f = 0
for i in ds:  #统计正确和错误文本个数
    if i[col - 1] == 1:
        t += 1
    else:
        f += 1
if t == 0:  #防止出现log(0)
    tt = 0
else:
    tt = t * 1.0 / len(ds)
    tt = -tt * math.log(tt, 2)
if f == 0:
    ff = 0
else:
    ff = f * 1.0 / len(ds)
    ff = -ff * math.log(ff, 2)
hd = tt + ff   #计算经验熵
```

```
a = set() #特征
b = {}    #每个特征的文本数
c = {}    #每个特征的正确文本数
for i in ds:
    a.add(i[p])
for i in a:  #初始化
    b[i] = 0
    c[i] = 0
for i in ds:  #统计
    b[i[p]] += 1
    if i[col - 1] == -1:
        c[i[p]] += 1
```

```
temp = b[i] * 1.0 / len(ds)  #特征的概率
if c[i] > 0:
    t = c[i] * 1.0 / b[i] #正确文本概率
else:
    t = 0
if b[i] - c[i] > 0:  #错误文本概率
    f = (b[i] - c[i]) * 1.0 / b[i]
else:
    f = 0
if t > 0:  #防log(0)处理
    tt = -t * math.log(t, 2)
else:
    tt = 0
if f > 0:  #防log(0)处理
    ff = -f * math.log(f, 2)
else:
    ff = 0
temp *= (tt + ff)
ig += temp   #计算经验熵
```

```
    gain = hd - ig  #计算增益

    if gain > max_ig:  #选最大值
        max_ig = gain
        max_col = p
return max_col  #返回最大增益列
```

然后是 C4.5 部分，增益的计算和之前相同，主要截图特征的熵和增益率的计算：

```
splitinfo = 0
for i in b:
    if b[i] == 0:  #防止除以0
        continue
    temp = b[i] * 1.0 / len(ds)  #特征的概率
    temp = -temp * math.log(temp, 2) #公式
    splitinfo += temp  #累加
```

```
    if gain == 0:
        rt = 0 #不考虑0
    else:
        rt = gain * 1.0 / splitinfo #计算增益率
    if rt > max_rt:  #选择最大增益率
        max_rt = rt
        max_col = p
return max_col  #返回最大增益率列
```

接下来是 CART，关键统计值的得出不再截图，主要截图计算部分：

```
for j in a:  #特征j的基尼系数累加（b是该特征文本数，c是正确文本数，相应b-c就是错误的个数）
    gini += b[j] * 1.0 / len(ds) * (1 - (c[j]*1.0/b[j]) * (c[j]*1.0/b[j]) - (b[j]-c[j])*1.0/b[j] * (b[j]-c[j])*1.0/b[j])
```

建树的过程 decision_tree：

```
flag = 1 #课件中的边界条件 (1)
for i in data_set: #如果全是1则返回1
    if i[col - 1] != 1:
        flag = 0
        break
if flag == 1:
    return 1
    flag = -1
for i in data_set: #如果全是-1则返回-1
    if i[col - 1] != -1:
        flag = 0
        break
if flag == -1:
    return -1
```

```
if max(choices) == 0: #如果特征集为空
    t = 0
    f = 0
    for i in data_set:
        if i[col - 1] == 1:
            t += 1
        else:
            f += 1
    if t > f:  #选择最多的类别返回
        return 1
    else:
        return -1
```

```
# 选择一种决策方法
# choice = id3(data_set, choices)
choice = c45(data_set, choices)
# choice = cart(data_set, choices)
```

```
tree['col'] = choice
choices[choice] = 0 #删除选中的特征
a = set()
b = {}
for i in data_set:  #统计特征取值
    a.add(i[choice])
for i in a:
    b[i] = []
for i in data_set: #收集每个取值的文本作为新的数据集
    b[i[choice]].append(i)

for i in b:  #递归建树
    if len(b[i]) == 0:
        tree[i] = 1
    else:
        tree[i] = decision_tree({}, b[i], choices)
choices[choice] = 1  #回溯，重新启用那个特征
return tree
```

计算验证集或测试集文本的类别 calc：

```
def calc(tree, data):
    chose = data[tree['col']] #获取当前节点对应的特征
    if tree.has_key(chose):  #确保有这个特征取值
        val = tree[chose]
    else:       #如果没见过这个取值就无法预测，直接猜1
        return 1

    if val == 1:    #如果已经是叶节点就返回
        return 1
    elif val == -1:
        return -1
    else:           #否则继续递归
        return calc(val, data)
    return 1
```

其余读取数据集以及统计部分基本与前几次相同。

4. 创新点&优化（如果有）

三、 实验结果及分析

1. 实验结果展示示例（可图可表可文字，尽量可视化）

采用小的数据集，和文档上的一样：

	Standard	Standard	Standard
1	1	1	1
2	0	1	-1
3	1	0	-1
4	0	0	-1

得出各项指数如下：

```
id3:
ent: 0.8112781245
col1: 0.5
col2: 0.5

c4.5:
splitinfo1: 1
splitinfo2: 1
col1: 0.5
col2: 0.5

gini:
col1: 0.25
col2: 0.25
[Finished in 0.0s]
```

打印建好的树，映射关系如下：

```
0 -1
1 {0: -1, 1: 1, 'col': 1}
col 0
[Finished in 0.0s]
```

均与手算结果一致。

2. 评测指标展示即分析（如果实验题目有特殊要求，否则使用准确率）

用 C4.5 做出来的验证集正确率约 62.7%，而其余两种决策方法均在 60.7%。

```
231     # choice = id3(data_set, choices)
232     choice = c45(data_set, choices)
233     # choice = cart(data_set, choices)
0.626666666667
[Finished in 0.3s]
```

四、 思考题

1. 决策树有哪些避免过拟合的方法？

答：（1）可以通过限制树的深度，在达到最大深度之后就停止建树；（2）可以采用更多的数据集来进行训练；（3）剪枝，如果将某个属性划分去掉之后，决策树在验证集上的正确率有提高，则删去这个划分。

2. C4.5 相比于 ID3 的优点是什么？

答：ID3 倾向于选择取值个数比较多的特征，例如本次实验的第一列算出来的信息增益就会比较大。而 C4.5 会除以该特征的熵。取值多的特征虽然增益大，可是熵也会相对较大，所以选择出来的特征会更加合理。在本次实验中，C4.5 在验证集中的正确率始终略高于 ID3。

3. 如何用决策树来判断特征的重要性？

答：决策树上越靠近根节点，也就是深度越低的节点所表示的特征越重要，因为它们的区分度是最高的。

五、 参考资料

[1] https://www.cnblogs.com/nxld/p/6371453.html 第一点第一行