

基于 ID3、C4.5 的决策树算法的机器学习

517030910227 洪瑄锐

0. 引言

H. Simon 曾于 1983 年给出了一个关于学习的哲学式说明：如果一个系统能够通过执行某种过程而改进其性能，这就是学习。机器学习是一门多领域交叉学科，专门研究计算机怎样模拟或实现人类的学习行为，以获取新的知识或技能，重新组织已有的知识结构使之不断改善自身的性能，当前机器学习正被应用于各个领域，它的重要性不言而喻。

在课堂上老师为我们介绍了机器学习中监督式学习的三种决策树算法：ID3, C4.5, CART，使之前没有接触过相关内容的我对机器学习有了一个最基本的了解，并在本次大作业中尝试对 ID3, C4.5 进行代码实现。

本论文共分为三个部分 1. 数据分析 2. 算法及代码分析 3. 输出结果分析 4. 感想。

1. 数据分析

1.1 数据来源与分析

本次大作业采用的数据集为 UCI 网站上的 breast-cancer-wisconsin 数据集。数据共有 10 个属性，1 个分类，训练和测试时采用 4-10 号属性，预测 11 号分类。

Attribute	Domain
1. Sample code number	id number
2. Clump Thickness	1 - 10
3. Uniformity of Cell Size	1 - 10
4. Uniformity of Cell Shape	1 - 10
5. Marginal Adhesion	1 - 10
6. Single Epithelial Cell Size	1 - 10
7. Bare Nuclei	1 - 10
8. Bland Chromatin	1 - 10
9. Normal Nucleoli	1 - 10
10. Mitoses	1 - 10
11. Class:	2 for benign, 4 for malignant

部分数据如图所示

```
1 1000025 5 1 1 1 2 1 3 1 1 2
2 1002945 5 4 4 5 7 10 3 2 1 2
3 1015425 3 1 1 1 2 2 3 1 1 2
4 1016277 6 8 8 1 3 4 3 7 1 2
5 1017023 4 1 1 3 2 1 3 1 1 2
6 1017122 8 10 10 8 7 10 9 7 1 4
7 1018099 1 1 1 1 2 10 3 1 1 2
8 1018561 2 1 2 1 2 1 3 1 1 2
9 1033078 2 1 1 1 2 1 1 1 5 2
```

1.2 交叉验证

在进行决策树构建时，会因数据的特别性，算法的局限性等原因造成误差，出现过拟合或欠拟合。因此对建立的决策树进行验证是不可或缺的步骤。交叉验证是将训练数据中的小部分代入决策树，得到决策树的评价指标的方法。

本次大作业在交叉验证时，输入 590 个数据作为训练数据，98 个数据作为测试数据。在决策树构建完成后通过输入属性值得到预测结果，与真实结果比较后得到学习效果和推广性能。

2. 算法及代码分析

2.1 ID3 算法

2.1.1 ID3 算法简介

ID3 算法是决策树中最简单的算法，其主要是通过信息增益来进行特征的选择，根据公式选择最优子项继续向下分裂来建树。ID3 算法的核心是采用贪婪策略，通过信息增益来计算下一个分类目标，而信息增益的核心为信息熵，熵起源于物理学，是上世纪五十年代由香农引进用来表示不确定量。信息熵是影响信息增益变化的关键，信息熵越低就表示整个系统会更加的有序。

计算公式：

1、信息熵(Entropy)

$$Entropy(S) = - \sum_{i=1}^m p(u_i) \log_2 p(u_i)$$

其中， $p(u_i) = \frac{|u_i|}{|S|}$ ， $p(u_i)$ 为类别在样本中出现的概率。

2、信息增益(Information gain)

$$infoGain(S, A) = Entropy(S) - \sum_{V \in Value(A)} \frac{|S_V|}{|S|} Entropy(S_V)$$

其中， A 表示样本的属性， $Value(A)$ 是属性所有的取值集合。 V 是 A 的其中一个属性值， S_V 是 S 中 A 的值为 V 的样例集合。

2.1.2 ID3 主要代码分析（绿色注释部分即代码主要思想）

（1）计算信息熵

//根据属性的取值，计算该属性的熵

```
double compute_entropy(vector<unsigned> v) //输入为不同组数据中某种属性或分类的取值集合，例如classes: (2, 2, 4, 4, 2, 2, 4, 4, 2.....)
{
    vector<unsigned> unique_v;
    unique_v = unique(v); //去掉重复元素，在本数据集中即 (2, 4)

    vector<unsigned>::iterator itr;
    itr = unique_v.begin();

    double entropy = 0.0;
    auto total = v.size();
    while (itr != unique_v.end()) //遍历classes的每种取值，计算 概率*log2(概率)，取负后加入entropy
    {
        double cnt = count(v.begin(), v.end(), *itr); //计算每种classes取值在集合中的个数
        entropy -= cnt / total * log2(cnt / total);
        itr++;
    }
    return entropy;
}
```

（2）计算信息增益

```
vector<double> compute_gain(vector<vector<unsigned> > truths) //输入为训练集
{
    vector<double> gain(truths[0].size() - 1, 0); //存储各个属性的信息增益，规模数为truths的size-1是因为训练集各个数据的最后一列为classes，无需计算
    vector<unsigned> attribute_vals; //用来针对每个属性存储其取值，得到该属性在各个数据中的取值集合
    vector<unsigned> labels; //用来存储classes的取值
    for (unsigned j = 0; j < truths.size(); j++) //将classes取值放入labels
    {
        labels.push_back(truths[j].back());
    }

    for (unsigned i = 0; i < truths[0].size() - 1; i++) //遍历每一种属性
    {
        for (unsigned j = 0; j < truths.size(); j++) //将每个数据j中的第i种属性的取值放入attribute_vals
            attribute_vals.push_back(truths[j][i]);

        vector<unsigned> unique_vals = unique(attribute_vals); //去重，得到第i种属性的所有可能取值
        vector<unsigned>::iterator itr = unique_vals.begin();
        vector<unsigned> subset; //subset存储对应于数据的第i种属性为某个取值时的classes集合
        while (itr != unique_vals.end()) //遍历第i种属性的所有可能取值
        {
            for (unsigned k = 0; k < truths.size(); k++)
            {
                if (*itr == attribute_vals[k])
                {
                    subset.push_back(truths[k].back()); //将数据truth[k]的第i种属性取*itr时的classes放入subset
                }
            }
            double A = (double)subset.size();
            gain[i] += A / truths.size() * compute_entropy(subset); //得到第i种属性取*itr的概率以及该属性取值对应classes的信息熵，加到H(D|A)中
            itr++; //计算第i种属性的下一个取值
            subset.clear();
        }
        gain[i] = compute_entropy(labels) - gain[i]; //g(D,A)=H(D)-H(D|A)
        attribute_vals.clear();
    }
    return gain;
}
```

（3）构建决策树

```

struct Tree {
    unsigned root;//节点属性值
    vector<unsigned> branches;//节点可能取值
    vector<Tree> children; //孩子节点
};

```

```

//递归构建决策树

```

```

void build_decision_tree(vector<vector<unsigned> > examples, Tree &tree)
{
    //递归终止条件: 判断所有实例是否都属于同一类, 如果是, 则决策树是单节点
    vector<unsigned> labels(examples.size(), 0);
    for (unsigned i = 0; i < examples.size(); i++)
    {
        labels[i] = examples[i].back();
    }
    if (unique(labels).size() == 1)
    {
        tree.root = labels[0];
        return;
    }

    //递归终止条件: 判断是否还有剩余的属性没有考虑, 如果所有属性都已经考虑过了,
    //那么此时属性数量为0, 将训练集中最多的类别标记作为该节点的类别标记
    if (count(examples[0].begin(), examples[0].end(), 110) == examples[0].size() - 1)//只剩下一列类别标记
    {
        tree.root = find_most_common_label(examples);
        return;
    }
    //计算信息增益, 选择信息增益最大的属性作为根节点, 并找出该节点的所有取值

    vector<double> standard = compute_gain(examples);

    //要是采用C4.5, 将上面一行注释掉, 把下面一行的注释去掉即可
    //vector<double> standard = compute_gain_ratio(examples);

    tree.root = 0;
    for (unsigned i = 0; i < standard.size(); i++)
    {
        if (standard[i] >= standard[tree.root] && examples[0][i] != 100)
            tree.root = i;
    }
}

```

```

//计算信息增益, 选择信息增益最大的属性作为根节点,并找出该节点的所有取值

vector<double> standard = compute_gain(examples);

//要是采用C4.5, 将上面一行注释掉, 把下面一行的注释去掉即可
//vector<double> standard = compute_gain_ratio(examples);

tree.root = 0;
for (unsigned i = 0; i < standard.size(); i++)
{
    if (standard[i] >= standard[tree.root] && examples[0][i] != 100)
        tree.root = i;
}

tree.branches = find_attribute_values(tree.root, examples);
//根据节点的取值, 将examples分成若干子集
vector<vector<unsigned> > new_examples = drop_one_attribute(tree.root, examples);
vector<vector<unsigned> > subset;
for (unsigned i = 0; i < tree.branches.size(); i++)
{
    for (unsigned j = 0; j < examples.size(); j++)
    {
        for (unsigned k = 0; k < examples[0].size(); k++)
        {
            if (tree.branches[i] == examples[j][k])
                subset.push_back(new_examples[j]);
        }
    }
    //对每一个子集递归调用build_decision_tree()函数
    Tree new_tree;
    build_decision_tree(subset, new_tree);
    tree.children.push_back(new_tree);
    subset.clear();
}
}

```

2.2 C4.5 算法

2.2.1 C4.5 算法简介

C4.5 与 ID3 不同的是它采用信息增益率选择属性。

计算公式：

- 信息增益率 $g_R(D, A)$ 定义为其信息增益 $g(D, A)$ 与训练数据集 D 关于特征 A 的值的熵 $H_A(D)$ 之比，即

$$g_R(D, A) = \frac{g(D, A)}{H_A(D)}$$

- 其中分裂信息为

$$H_A(D) = - \sum_{i=1}^n \frac{|D_i|}{|D|} \log \frac{|D_i|}{|D|}$$

- 在这些公式中， D_1, D_2, \dots, D_n 是根据属性 A 对样本集分割形成的 n 个样本子集，每个子集中的样本属性 A 取值相同。

2.2.2 C4.5 主要代码分析

(1) 计算信息增益率

//计算数据集中所有属性的信息增益率

```
vector<double> compute_gain_ratio(vector<vector<unsigned> > truths)
{
    vector<double> gain = compute_gain(truths); //获得每种属性的信息增益熵
    vector<double> entropies;
    vector<double> gain_ratio;

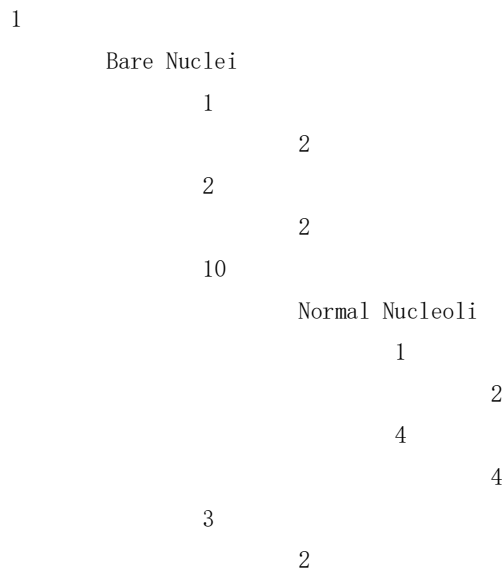
    for (unsigned i = 0; i < truths[0].size() - 1; i++) //最后一列是类别标签, 不需要计算信息增益率
    {
        vector<unsigned> attribute_vals(truths.size(), 0);
        for (unsigned j = 0; j < truths.size(); j++)
        {
            attribute_vals[j] = truths[j][i];
        }
        double current_entropy = compute_entropy(attribute_vals);
        if (current_entropy)
        {
            gain_ratio.push_back(gain[i] / current_entropy);
        }
        else
            gain_ratio.push_back(0.0);
    }
    return gain_ratio;
}
```

3. 输出结果分析

3.1 输出决策树如下所示 (2 或 4 为预测结果)

ID3:

Uniformity of Cell Shape



	5		
		Bland Chromatin	
		1	
			2
		2	
			4
		5	
			2
	4		
		2	
4			
		Single Epithelial Cell Size	
	7		
		2	
	2		
		Normal Nucleoli	
		6	
			4
		1	
			2
		5	
			4
		3	
			4
		2	
			2
	5		
		4	
	8		
		4	
	3		
		4	
	6		
		Normal Nucleoli	
		4	
			4
		5	
			4
		3	
			2
	4		
		Normal Nucleoli	
		8	
			2

			7	
				2
			10	
				4
			1	
				2
			9	
				4
		10		
			4	
		1		
			2	
8				
	Bland Chromatin			
		3		
			Mitoses	
				1
				2
				3
				4
		2		
			4	
		8		
			4	
		5		
			4	
		7		
			4	
		4		
			4	
		9		
			4	
		10		
			4	
10				
	4			
2				
	Bare Nuclei			
		1		
			2	
		10		
			4	
		2		
			Mitoses	

		10	
			4
		1	
			2
		2	
			2
3			
	2		
4			
	Bland Chromatin		
		1	
			2
		4	
			4
6			
	4		
5			
	2		
3			
	Bare Nuclei		
	3		
		4	
	7		
		4	
	1		
		2	
	4		
		4	
	10		
	Marginal Adhesion		
		1	
			4
		10	
			4
		7	
			4
		3	
			4
		2	
			4
		5	
			2
		6	
			4

5	2		
		2	
	5		
		4	
	8		
		4	
	Bare Nuclei		
	9		
		4	
	7		
		4	
	1		
		4	
6	8		
		4	
	2		
		4	
	3		
		Normal Nucleoli	
		10	
			4
		6	
			2
	10		
		4	
	5		
7		4	
	4		
		4	
		4	
	Bare Nuclei		
	1		
		Single Epithelial Cell Size	
		6	
			4
		3	
			2
		2	
8			2
	7		
		Normal Nucleoli	
9		8	
			2

			10	
				4
	5			
		4		
	4			
		4		
	8			
		4		
	10			
		4		
7				
	Normal Nucleoli			
	1			
		4		
	4			
		Bare Nuclei		
		5		
			4	
		8		
			2	
		10		
			4	
	8			
		4		
	10			
		4		
	7			
		4		
	9			
		4		
	6			
		4		
	3			
		4		
	2			
		2		
	5			
		4		
9				
	4			

C4.5

Bare Nuclei

1

Mitoses

1

Normal Nucleoli

1

Marginal Adhesion

1

2

3

2

2

2

6

Single Epithelial Cell Size

10

4

3

2

4

2

5

4

3

Uniformity of Cell Shape

6

Single Epithelial Cell Size

6

4

3

2

5

4

1

2

2

2

4

4

3

2

9

				4	
		10		4	
		2		2	
		8			
			Uniformity of Cell Shape		
				1	
					2
				10	
					4
		6			
			2		
		5			
			4		
5					
	2				
2					
	2				
4					
	4				
8					
	4				
10					
	4				
3					
		Normal Nucleoli			
		10			
			4		
		2			
			2		
10					
	Bland Chromatin				
	3				
		Marginal Adhesion			
		5			
			2		
		1			
			Mitoses		
				1	
					2
				2	
					4
		10			

			4
		6	
			4
		8	
			4
		3	
			4
		2	
			4
		9	
			4
		4	
	9		
		4	
	4		
		4	
	5		
		4	
	7		
		4	
	6		
		4	
	1		
		4	
	10		
		4	
	8		
		4	
	2		
		4	
2			
	Single Epithelial Cell Size		
	2		
		Uniformity of Cell Shape	
		1	
			2
		5	
			4
		2	
			2
		3	
			2
	5		

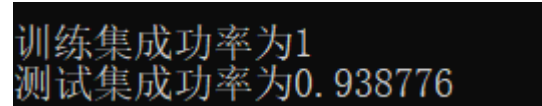
Feature	Score
Mitoses	4
	3
	10
	4
	1
	2
	10
	4
	6
	4
	4
	4
Bland Chromatin	4
	3
Uniformity of Cell Shape	8
	2
	3
	4
	4
	4
	4
	8
	4
	7
	4
	1
	2
Uniformity of Cell Shape	3
	4
	1
	2
	8
	4
	5
Normal Nucleoli	10
	4
	6

			2
	4		
		Bland Chromatin	
		3	
			4
		5	
			4
		2	
			2
	2		
		2	
	7		
		4	
	9		
		4	
9			
	4		
7			
	Marginal Adhesion		
	3		
		4	
	4		
		4	
	9		
		2	
	7		
		4	
	5		
		2	
5			
	Uniformity of Cell Shape		
	7		
		4	
	6		
		4	
	5		
		4	
	4		
		Normal Nucleoli	
		5	
			4
		7	
			2
		3	

			2
1			
	Single Epithelial Cell Size		
		2	
			2
		1	
			4
8			
	4		
9			
	4		
3			
	4		
10			
	4		
2			
	2		
8			
	Normal Nucleoli		
	8		
		4	
	3		
		4	
	1		
		4	
	4		
	Uniformity of Cell Shape		
		6	
			4
		7	
			2
10			
	4		
7			
	4		
9			
	4		
6			
	4		
2			
	2		
6			
	4		

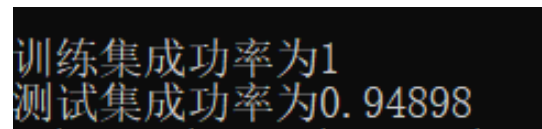
3.2 学习效果和推广性能

ID3



训练集成功率为1
测试集成功率为0.938776

C4.5



训练集成功率为1
测试集成功率为0.94898

由输出结果可知，两种决策树的学习性能和推广效果均较好，且 C4.5 优于 ID3。但这两种算法均存在缺点，比如 ID3 算法会优先选取具有大量值的属性，只能处理离散型属性，C4.5 作为 ID3 的改进，虽然解决了上述两种问题，但存在在构造树的过程中，需要对数据集进行多次的顺序扫描和排序，因而导致算法的低效，当数据量较大时效率会很低。

4. 感想

这次的大作业让我有机会动手实现两种决策树的算法，亲身感受了机器学习最基础部分的实现与使用计算机预测的神奇。但是我仍有许多不足之处，仍有许多算法可以改进决策树的性能，在日后的学习中我会不断完善和改进。希望这成为我进入机器学习领域，扩展知识的开始。