机器学习——决策树实验报告

第1节　多种决策树算法实验对比

首先，我使用python语言，不借助软件包，自己代码实现了ID3、C4.5与CART三种决策树生成算法。在数据集的选择上，我从UCI上选择了Car Evaluation数据集[1] [3]进行实验比较。

第1.1小节　数据集简介

Car Evaluation数据集中包含了对于汽车不同特征的离散描述以及对于的汽车评价标签，其中包括6项特征：

1. **buying**: vhigh, high, med, low.
2. **maint**: vhigh, high, med, low.
3. **doors**: 2, 3, 4, 5more.
4. **persons**: 2, 4, more.
5. **lug_boot**: small, med, big.
6. **safety**: low, med, high.

对应的标签为：

- **Values**: unacc, acc, good, vgood.

[1] http://archive.ics.uci.edu/ml/datasets/Car+Evaluation

第1.2小节　决策树算法实验对比

Car Evaluation数据集一共包括1728条数据，我将这些数据按照4:1划分成训练集和测试集，训练集用于生成决策树模型，测试集用于评价模型预测的准确率，最终未经过剪枝的三种算法在测试集上的准确率表现如下表1所示：

表 1: 不同决策树生成算法在Car Evaluation数据集上的准确率

生成算法	ID3	C4.5	CART
准确率	0.9017	0.9017	**0.9624**

其中CART算法在测试集上的表现准确率最高，并且相较于其他两种决策树算法的表现，准确率的提高程度比较大，而后ID3算法与C4.5算法的准确率相差无几。另外，我还对比了三种算法在建树和预测的耗时，其中使用训练集1382条数据用于建树，测试集346条数据用于测试，如下表4所示：

表 2: 不同决策树生成算法建树和预测的耗时

生成算法	ID3	C4.5	CART
建树时间(ms)	**54.06**	58.28	73.79
预测时间(ms)	1.57	**1.50**	5.46

其中ID3算法和C4.5算法的建树耗时和预测耗时没有明显的差距，然而由于CART算法生成的决策树是二叉树，此算法所得到的决策树的生成和预测均会更加耗时。

总体而言，相较于ID3算法，C4.5算法引入信息增益率来作为选择分裂特征的标准，消除了ID3算法对于取值数目多的特征的偏好，在本数据集中未能表现出来的C4.5的其他优点还包括 [1]：

1. 能够对连续属性进行离散化处理；
2. 能够处理不完整数据；

但C4.5算法同时也有其他未展露的缺点包括 [1]：

1. C4.5算法只能用于分类；
2. C4.5使用的熵模型拥有大量耗时的对数运算，连续值还有排序运算，所以只适合于能够驻留于内存的数据集，当训练集大得无法在内存容纳时，程序无法运行；

CART算法相较于C4.5算法和ID3算法取得了更高的准确率，作为trade-off，其生成树和预测的耗时都相应地增加了。CART算法的其他有点还包括：

1. 可以解决回归问题；

2. 能够处理连续型数据；

但CART算法同时也有其他缺点包括 [2]：

1. 每次用最优特征进行划分，这种贪心算法很容易陷入局部最优；

2. 具有样本敏感性，样本的一点改动足以影响整个树的结构；

更直观地，我还随机采样出了40行数据用于对ID31、C4.52、CART3这三种决策树生成算法所生成的树做可视化对比。

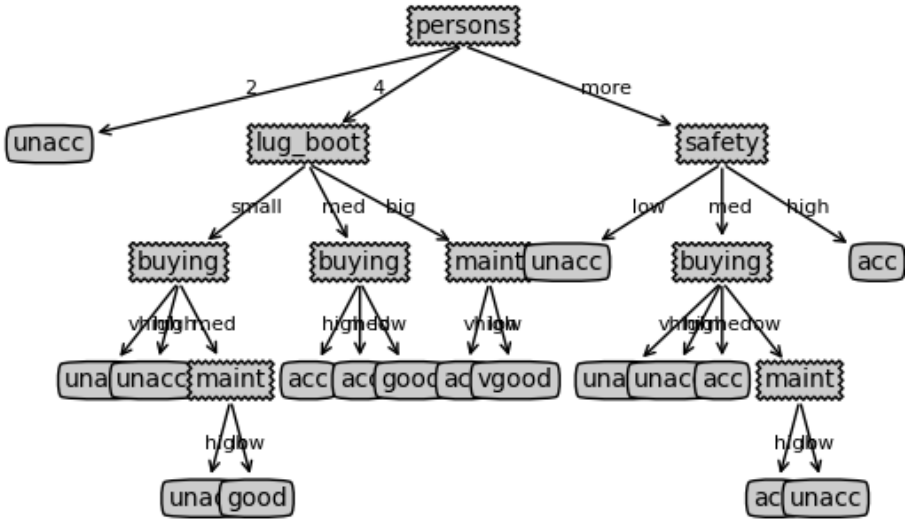

图 1: ID3算法生成的决策树

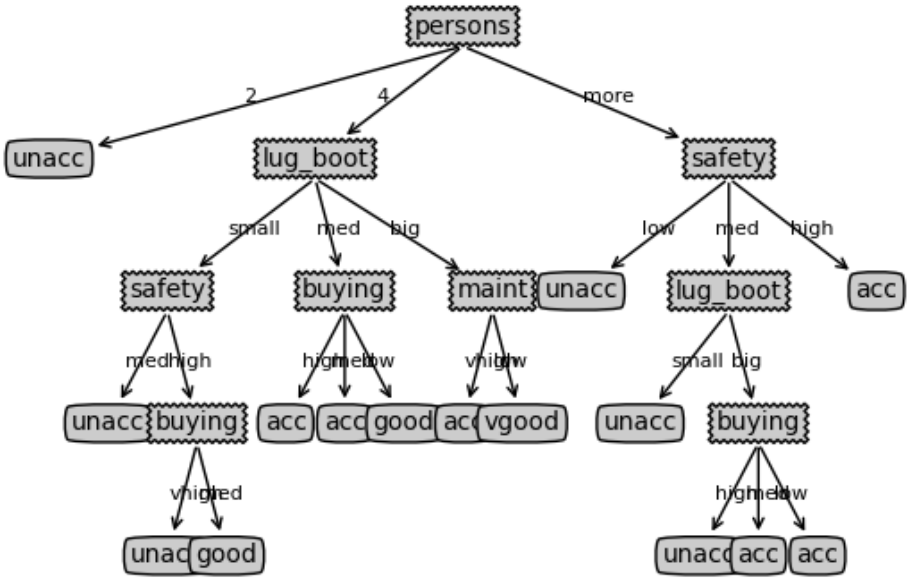

图 2: C4.5算法生成的决策树

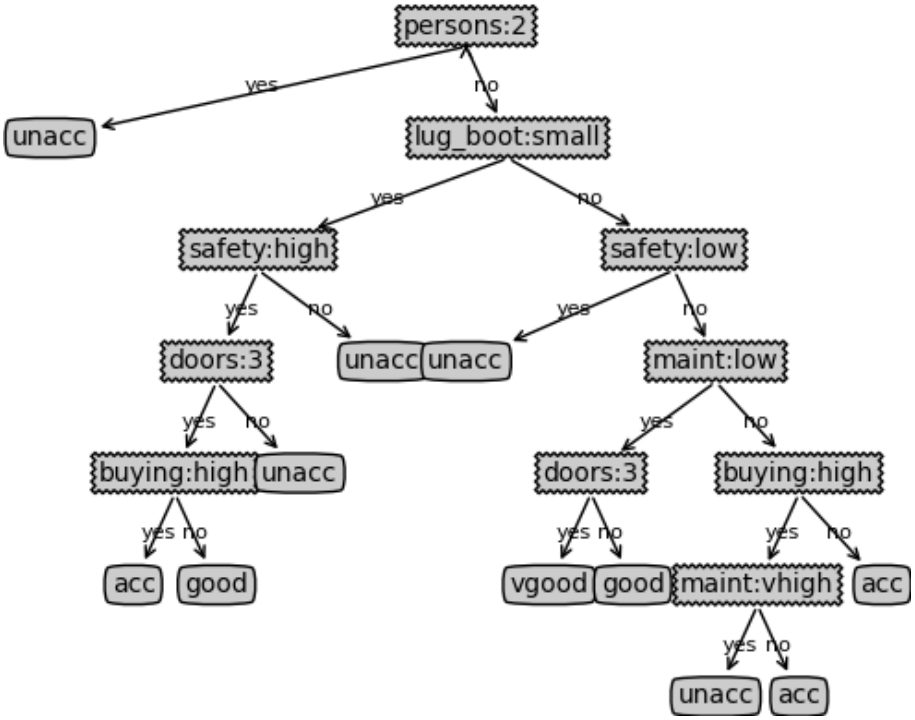

图 3: CART算法生成的决策树

通过图1与图2的对比可以很直观地发现，ID3树更倾向于选择取值数目多的特征作为分裂的特征，如ID3树从根结点开始的第三层，ID3算法中选取了三次“buying”特征作为分裂的特征，正是因为“buying”特征相较于“safety”和“lug_boot”特征有更多的取值。而图3与图1、图2的比较就更为明显，CART算法的生成树是一棵二叉树，而ID3、C4.5算法的生成树的一棵多叉树。

第2节　C4.5算法上不同剪枝算法的比较

在C4.5算法上，我实现了最小误差剪枝(Minimum Error Pruning, MEP)算法 [5]，并与原始的基于经验熵的剪枝算法进行比较，同样基于Car Evaluation数据集进行实验。

第2.1小节　最小误差剪枝算法

MEP算法基于自底向上的方式，递归地从树的叶结点向上回缩。对于树中每个非叶结点T，首先使用公式1计算该结点的误差$E_r(T)$:

$$E_r(T) = \frac{N(T) - N_c(T) + (k-1)}{N(T) + k} \tag{1}$$

其中$N(T)$表示该结点T所包含的实例总数，$N_c(T)$为T中实例数最多的主类的实例数量，k为分类任务的分类数量。然后计算该结点每个分枝$t \in T$的误差$E_r(t)$，之后按照公

式2加权相加：

$$E_r\prime(T) = \sum_{t \in T} \frac{N(t)}{N(T)} E_r(t) \tag{2}$$

而后比较$E_r(T)$与$E_r\prime(T)$，如果$E_r(T) \leq E_r\prime(T)$则进行裁剪；否则保留该子树。重复以上剪枝过程，直至不能继续为止。

第2.2小节　MEP算法与原始剪枝算法的比较

因为原始基于经验熵的剪枝算法存在一个参数α来控制叶节点经验熵和模型复杂程度，而MEP算法没有需要调节的超参数，表3展示了不同α的取值下，原始的剪枝算法和MEP剪枝算法在测试集上的表现。

表 3: MEP算法与原始剪枝算法的比较

剪枝算法	基于经验熵的剪枝算法				MEP	剪枝前
α	1.0	1.5	2.0	2.5	–	–
准确率	0.9017	**0.9191**	0.9162	0.8988	0.8872	0.9017

从表3可以发现，基于经验熵的剪枝算法，通过调整α的取值可以使得C4.5决策树生成算法在测试集上的准确率有所提高，且不同α对剪枝效果具有一定程度的影响，然而经过MEP算法的剪枝，模型相较于剪枝前的准确率却稍微下降了。我继续调整训练集和测试集的比例对基于经验熵的剪枝算法和MEP剪枝算法的表现做了进一步的比较，此时α取1.5。

表 4: MEP算法在不同数据量的数据集上的表现

训练集占比	20%	40%	60%	80%
剪枝前	0.8604	0.9036	0.9075	0.9017
基于经验熵的剪枝算法	0.8539	**0.9016**	**0.9133**	**0.9191**
MEP剪枝算法	**0.8612**	0.8833	0.8945	0.8872

可以看到，当训练集仅占完整数据集20%时，MEP剪枝算法在测试集上得到的准确率要高于剪枝前和基于经验熵的剪枝算法，由此说明MEP算法可能在小样本训练上存在一定的优势。

另外，我也随机采样出了40行数据对基于经验熵的剪枝算法4和MEP剪枝算法5所得到的决策树进行了可视化对比。

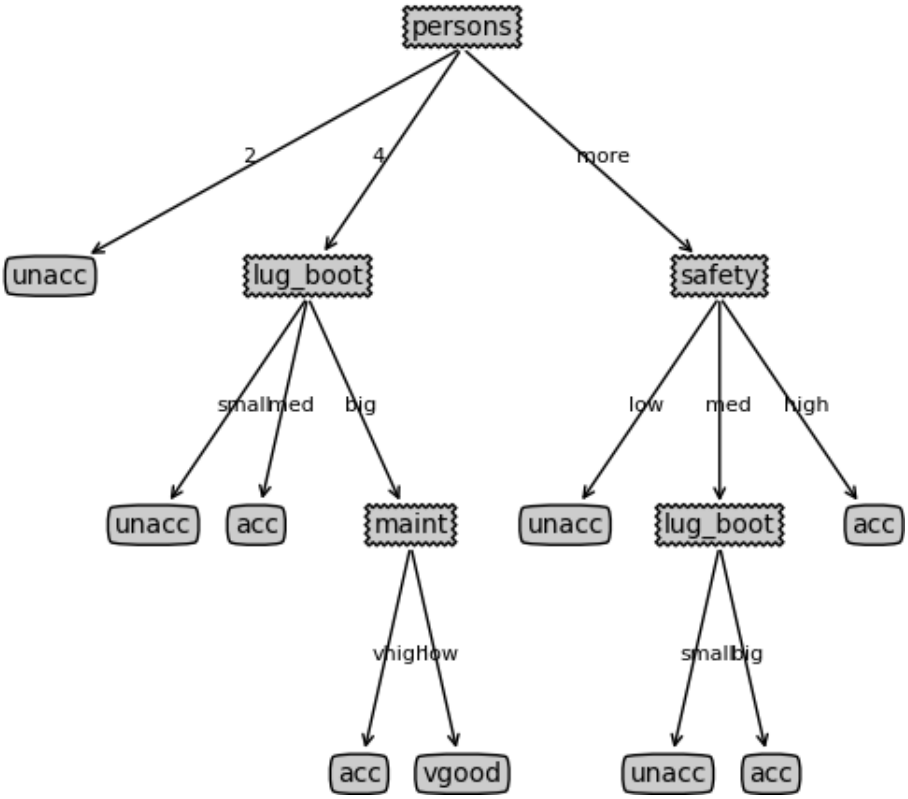

图 4: 基于经验熵剪枝之后的决策树

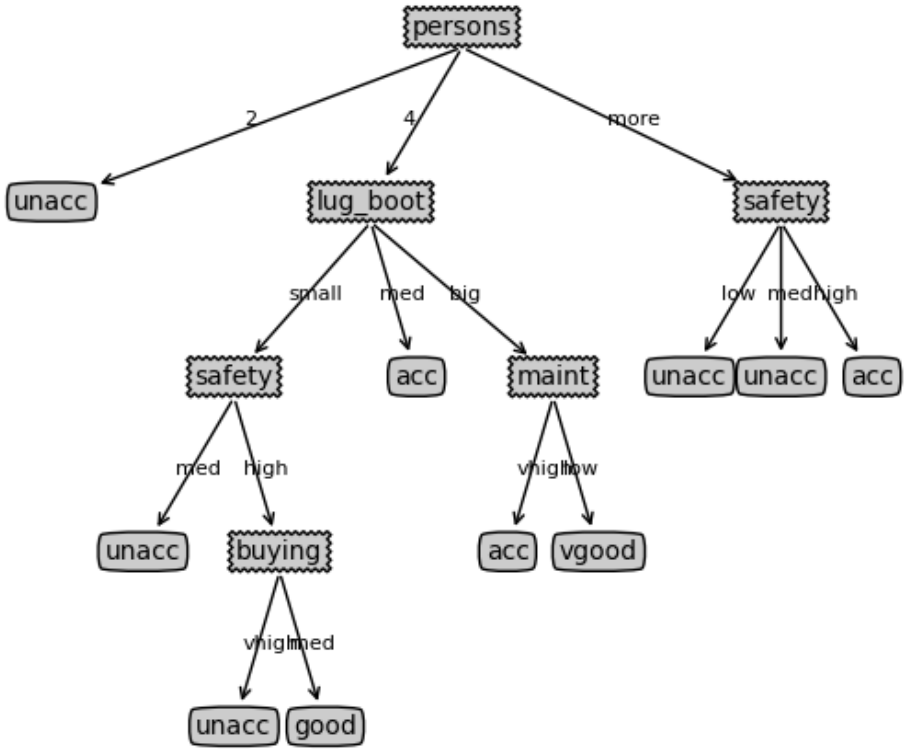

图 5: MEP算法剪枝之后的决策树

如图4和图5所示，MEP算法剪枝之后的决策树与原始剪枝算法所得到的决策树具有一定的区别，如MEP算法选择保留了从根结点开始第三层的“safety”子树，而基于经验熵的算法把此结点进行了剪枝。除此之外，已有文献也指出了MEP剪枝算法的缺点，如Cestnik和Biratko [4]指出MEP算法最主要的缺点就是$E_r(T)$的计算和训练样本的类数目k相关。

第3节　总结

在本作业中，我不借助软件包，自己代码实现了ID3、C4.5、CART三种决策树算法，对比了三种算法在Car Evaluation上的准确率、耗时表现，并作出可视化分析，此外我在C4.5算法上实现了MEP剪枝算法和原有的基于经验熵的剪枝算法，并且也对此两种剪枝算法进行实验比较和可视化分析。

参考文献

[1] 【机器学习】多种决策树算法比较. https://zhuanlan.zhihu.com/p/85731206.

[2] 决策树总结. https://www.cnblogs.com/jiangxinyang/p/9219771.html.

[3] M. Bohanec and V. Rajkovic. Knowledge acquisition and explanation for multi-attribute decision making. In 8th Intl Workshop on Expert Systems and their Applications, pages 59–78, 1988.

[4] B. Cestnik and I. Bratko. On estimating probabilities in tree pruning. In European Working Session on Learning, pages 138–150. Springer, 1991.

[5] T. Niblett and I. Bratko. Learning decision rules in noisy domains. In Proceedings of Expert Systems' 86, the 6th Annual Technical Conference on Research and development in expert systems III, pages 25–34, 1987.