成 绩 _____

北京航空航天大學
BEIHANG UNIVERSITY

使用决策树对 Iris 数据集进行分类

院（系）名称	高等理工学院
专 业 名 称	自动化
学 生 学 号	18376251
学 生 姓 名	乐祥立
指 导 教 师	秦曾昌

2021 年 5 月

使用决策树对 Iris 数据集进行分类

摘要

本文使用决策树算法实现了对 Iris 数据集[1]进行分类的任务，在随机输入的不同训练集和测试集下可以获得 90%以上的准确率，最高的准确率达到了 97.10%。在任务中还对连续数据的离散化方法、决策树的尺寸以及剪枝、决策树和朴素贝叶斯算法的对比以及软决策等方法进行了一定程度的探究。为提高决策树的稳定性和防止过拟合提供了一定的帮助。

关键词：决策树分类，朴素贝叶斯分类，Iris 数据集，决策树剪枝

一、引言

决策树是一种非常简单但是又非常有效果的机器学习算法。其名称的来源是因为算法决策分支画成的图形像一棵树的枝干，因此得名决策树。一个决策树主要由内部和外部的节点以及节点之间被称作枝干的内部连接组成。其中，内部节点是用于根据某一特征变量决定访问某一个子节点的，而外部节点是被称作叶子节点，是对输入数据分类的结果并且叶子节点不存在子节点。在机器学习中，决策树是一个预测模型，他代表的是对象属性与对象值之间的一种映射关系。

二、决策树原理和算法原理

决策树通过从树的根部到叶子节点对数据进行分类和整理。这种树状结构的分类器将输入数据所属的空间递归地划分成为互斥的子空间。按照这种结构，每一个训练数据都被认为是属于一个特定的子空间，该子空间可以通过一个标签，一个数值或者是一个动作来表达数据点的特征。决策树具有很好的透明度，我们可以根据决策树的结构轻松地解释一个决策是如何进行的。因此，当我们阐述决策树的条件规则时，可解释性得到了提高。

（一）决策树原理

1. 属性的信息熵

熵是物理学中用于描述无序性的变量，一个较大的熵意味着过程具有较大的随机性。在决策树算法中，树的结构是受到每一个属性的信息含量的启发性的指导的。熵用于描述每一个属性的信息从而作为一种分类的手段。

假设现在对于某一个属性而言有m个分类的类别，记为$C_i(\mathrm{i}=1,2,3,\ldots,\mathrm{m})$。对于某一个具体的分类类别$C_i$，标注数据集中属于该类别的数据的比例为$p_i$，因此该属性的熵可以以下的公式表达：

$$Entropy=\sum_{i=1}^{m}-p_i\cdot log_2\,p_i$$

我们也可以认为熵是一个训练集纯度的测量，熵越大，代表着数据集中的数据越不纯，即数据并不能够确定具体属于某一种类别。

2. 属性的信息增益

属性的信息增益(Information Gain)可以被定义为：信息熵−条件熵。由于信息熵是代表随机变量的复杂度，也称为不确定度；而条件熵是代表在某一个特定条件下，随机变量的复杂度或者不确定度的。因此信息增益可以被理解为在某一个特定的条件下，信息复杂度（不确定度）的减小的程度。信息增益越大，则越有利于决策树进行分类。

现假设所有的样例数据有 S 个，根据属性 A 的值划分成为了v组子数据集，每个类别有数据S_v个，那么可以将信息增益定义为：

$$IG(S,A)=Entropy(S)-\sum_{v\in Values(A)}\frac{|S_v|}{|S|}Entropy(S_v)$$

可以看出，以信息熵作为抉择点的划分依据可以使得整个数据集的熵不断地减小。从而实现分类的效果。

（二）算法原理

1. 总述

决策树使用 python 进行实现，整个流程可以划分为：数据的读取以及训练集和测试集的划分、当前节点下最优属性的选择以及数据划分、决策树分支迭代判断结束条件、

决策树的图形化绘制以及决策树的分类效果评价。其中的困难点在于 Iris 数据集中的连续值进行离散化的方法、决策树的递归形式、存储方式以及决策树的图形绘制。

2. 连续数据的离散化方法

决策树中计算信息增益时，数据集是按照属性划分成为不同的子数据集的，从而可以计算信息熵，但是对于连续数据，我们需要一种合适的划分，将数据集离散化为两个或多个类别，从而可以计算。最简单的划分方法就是将数据集进行二分。

给定数据集 D 和连续属性 a，假设属性 a 在 D 上有 n 个不同的取值，首先获取这些取值并按照从小到大的顺序进行排列，得到$\{a^1, a^2, \dots, a^n\}$，对于划分点$t(t \in [a^1, a^n])$可以将数据集 D 划分为两个子数据集D^+和D^-，D^+代表着在属性 a 上所有大于划分点t的数据的集合；D^-代表着在属性 a 上所有不大于划分点t的数据的集合。由于 n 个数据按照大小顺序有效划分为两个非空子集最多存在 n-1 种情况，因此划分 t 可以构成集合为：

$$T_a = \left\{ \frac{a^i + a^{i+1}}{2} | 1 \le i \le n-1 \right\}$$

即使用$[a^i, a^{i+1}]$的中位数作为划分点，划分数据集为两个子数据集从而可以计算信息增益为：

$$IG(D,a) = \max_{t \in T_a} IG(D,a,t) = \max_{t \in T_a} \left(Entropy(D) - \sum_{\lambda \in \{+,-\}} \frac{|D_t^\lambda|}{|D|} Entropy(D_t^\lambda) \right)$$

上述公式的意思是对于划分点 t 划分的数据集计算信息增益，并选择使得$IG(D,a)$最大的划分点 t，得到分类的划分点和数据子集。在进行属性 a 的IG计算之后，还需要计算其他属性的IG，再最终确定当前节点分类的属性以及该属性的划分点和数据子集。

3. 决策树的递归方式

考虑到决策树的绘制使用的是 treePlotter.py 文件[2]，其数据存储的结构是字典，这里我编写的决策树也使用字典的方式存储数据，并使用递归的方式来生成和结束字典。示意图如下：

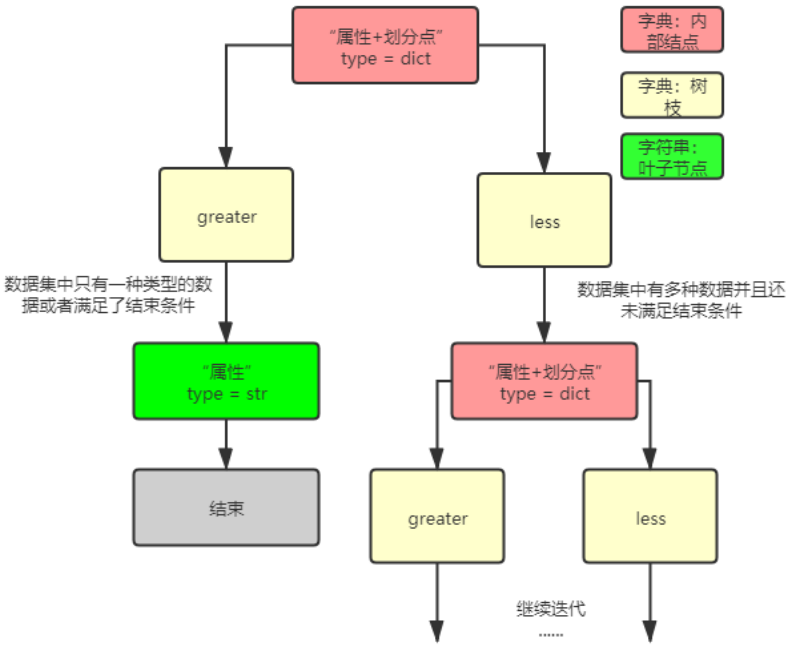


图 1 决策树使用字典的存储形式

4. 算法流程图

主程序算法：

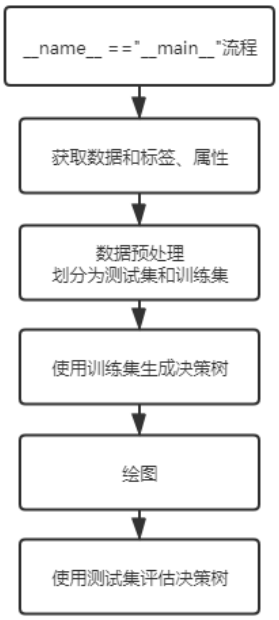


图 2 主程序算法

训练集生成决策树算法：

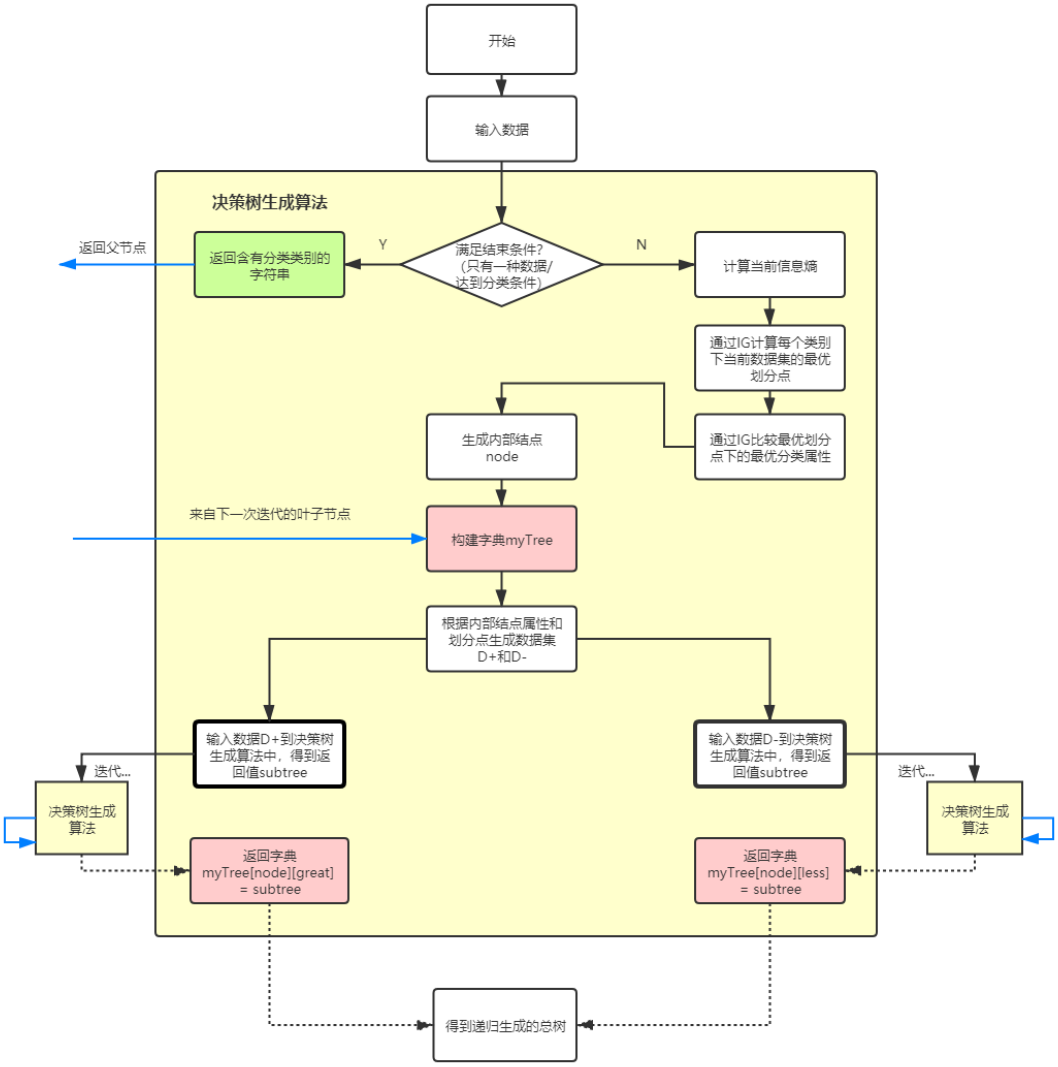


图 3 决策树生成算法

三、实验结果分析

（一）基本的决策树实现

1. 计算数据集的信息熵和第一个分类结点的最优选择

通过计算可以得到，对于整个数据集，每一个属性的信息熵均是：

$$Entropy(S) = 1.58457$$

在第一次划分的过程的当中，数据集采用 6：4 进行训练集和测试集的划分，设置排序随机种子为 1824。可以计算出各个类别的 Information Gain 和划分点 t 为

表 1 决策树第一次分类指标值

Attributes	Sepal Length	Sepal Width	Petal Length	Petal Width
Information Gain	0.49323	0.31291	**0.91830**	0.91830
划分点 t	5.75	3.95	**2.35**	0.8

可以看出对于原始数据集，Petal Length 和 Petal Width 均具有较大的信息增益，按照算法我们应当选取这两个属性其中的一个作为第一次的决策依据。根据代码实际选择的是首先出现的最大信息增益所对应的属性，即 Petal Length。

计算得到的结果和数据集中的文件 iris.names 中记录的 Petal Length 和 Petal Width 具有较大的分类相关性的说法一致。

2. **生成决策树**

采用和上述相同条件下的比例和随机数种子，可以得到运算的结果为

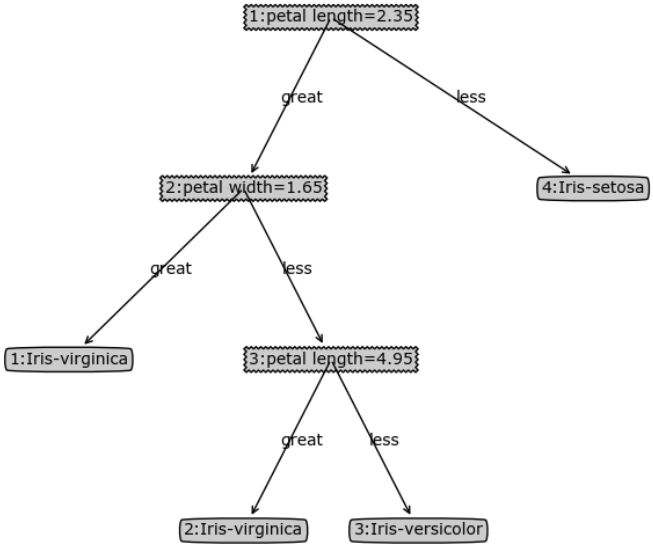


图 4 决策树示意图

第一次划分中将所有的 Iris-setosa 数据已经分类出来，生成 4 号叶子结点，共有 30 个数据。在第二次划分的过程中，使用的是 Petal Width 作为分类属性，划分点为 1.65。

表 2 决策树第二次分类指标值

Attributes	Sepal Length	Sepal Width	Petal Length	Petal Width
Information Gain	0.14892	0.06035	0.70796	**0.75828**
划分点 t	6.65	2.65	4.75	**1.65**

本次分类中，生成了 1 号叶子节点，共有 30 个 Iris-virginica 数据，并没有将剩余的

Iris-virginica 数据和 Iris-versicolor 数据分开。第三次分类中使用的是 Petal Length 对数据进行分类，划分点为 4.95。

表 3 决策树第三次分类指标值

Attributes	Sepal Length	Sepal Width	Petal Length	Petal Width
Information Gain	0.11901	0.05585	**0.46900**	0.13128
划分点 t	7.05	2.25	**4.95**	1.45

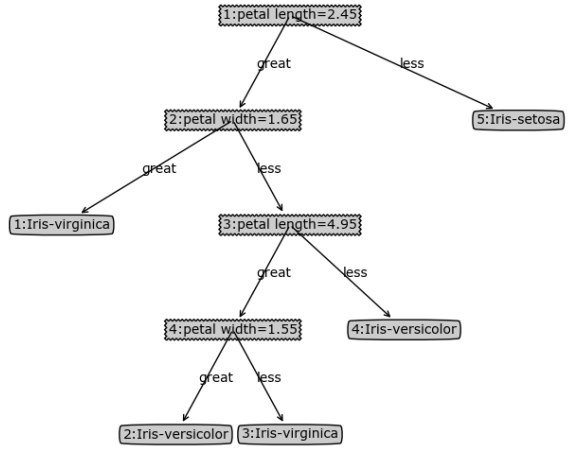
最后一次分类中将剩余的两种数据分开，分别是 3 个 Iris-virginica 和 27 个 Iris-versicolor。采用测试集对该决策树进行评估可以发现，总体的准确率为 95.65%，其中对 Iris-setosa 和 Iris-virginica 的分类完全正确，对 Iris-versicolor 的预测准确率只有 86.96%。

（二）决策树的复杂度和准确率

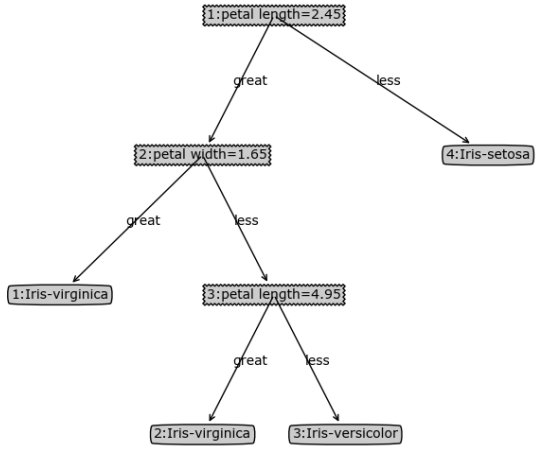
决策树在分类的过程中，其规则都是互斥并且完备的。即对于某一训练集而言，经过一些列规则的划分，总可以将每一个数据通过一层或者多层结点划分出来。因此可以在训练集上取得较好的效果。但是决策树在测试集上并不一定会取得理想的效果。此外，过多的分类，即分支，会使得抉择树的复杂度上升，也不利用提高分类器的泛化能力，因此需要适当约束抉择树的规模，防止过拟合的问题。

常用的约束决策树规模的方法有预剪枝和后剪枝。预剪枝就是在构造决策树的同时进行剪枝，一般是给定信息增益的阈值，当某次分类时的信息增益小于该阈值时则应当停止创建分支，避免过拟合，但是容易出现欠拟合。但是在实际运用中，这种方法因为难以确定阈值而不便于使用。后剪枝的过程是删除一些子树，用其叶子节点进行代替，通常是在决策树构造完成之后进行。剪枝的过程是对拥有同样父节点的一组节点进行检查，判断如果将其合并，熵的增加量是否小于某一阈值。如果确实小，则这一组节点可以合并一个节点。后剪枝的决策树欠拟合风险较小，泛化性能往往优于预剪枝决策树[3]。

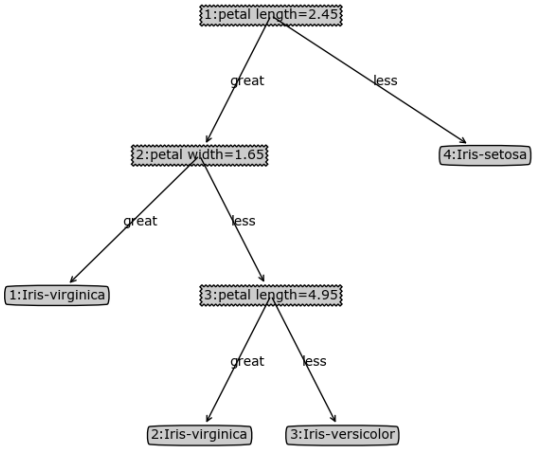
在本文中，我采用的方法是设定数据分类最小的阈值，即当进行一次分类时，若数据集中最小数量的数据小于设定的阈值threshold时，即不再进行分类，避免出现过拟合的现象。采用和前述问题中相同的数据比例 6：4，设定随机数种子为 12，当阈值threshold分别为 0，1，2 和 3 时，可以得到决策树和分类的准确率分别如下所示



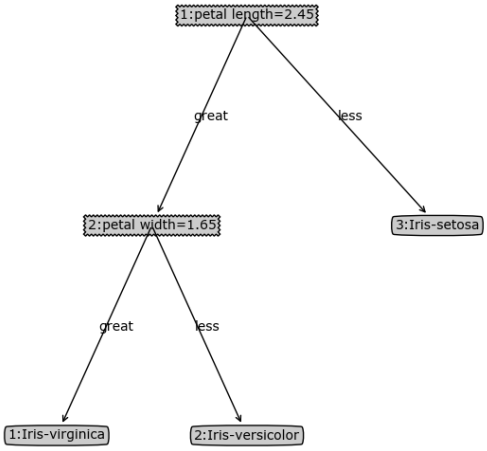
(a) Threshold = 0



(b) Threshold = 1



(c) Threshold = 2



(d) Threshold = 3

图 5 不同程度剪枝下的决策树

表 4 不同程度剪枝下的测试集准确率

Threshold	0	1	2	3
测试集准确率	95.43%	97.10%	97.10%	95.43%

可以看到，对决策树没有进行任何限制时，决策树的分支较多，部分叶子节点所包含的训练数据较少，出现了过拟合的现象。使用阈值进行一定程度上的限制之后，过拟合现象得到了抑制，准确率提升到了 97.10%。此时，若果再增大阈值，即所剩数据相对较多时也不进行分类，可以看到，此时决策树的分支数量已经达到了最小值，准确率和未约束时基本相同。

因此，决策树的大小和预测的准确率是存在着一个平衡的，若不对决策树进行约束，会在训练集上达到最优分类，但是会出现过拟合使得测试集上的分类准确率下降；适当的约束可以提高准确率；若进行较大的约束，即过多的剪枝，也会使得决策树预测的准

确率下降。

四、比较和改进

（一）朴素贝叶斯（Naïve Beyes）分类算法[4]

1. 算法原理

P(A|B)代表在事件 B 发生的条件下，A 发生的概率，称为事件 B 发生下事件 A 发生的条件概率。应用全概率公式可以得到贝叶斯定理为：

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

如果将 A 视为数据的特征 X，B 视为数据的类别 Y，则可以将公式转化为：

$$P(Y_k|X) = \frac{P(X|Y_k)P(Y_K)}{\sum_i P(X|Y_i)P(Y_i)}$$

因此通过上述公式就可以在已知数据的特征为 X 的情况下求解数据属于类别$\mathrm{Y_k}$的概率，通常情况下，我们可以从样本中国学习到先验概率$\mathrm{P(Y_k} = c_k)$，$(\mathrm{k} = 1,2,\dots,\mathrm{k})$，$\mathrm{c_k}$代表特征的分类类被。若可以求得条件概率分布$P(X|Y_k) = P(X_1 = x_1, X_2 = x_2, \dots, X_n = x_n|Y_k = c_k)$即可以计算出数据类别的概率。这里认为数据特征 X 的 n 个维度相互独立，即朴素贝叶斯假设，既可以将条件概率化简为：

$$P(X|Y_k) = \prod_{i=1}^{n} P(X_i = x_i|Y_k = c_k)$$

代入上述公式，考虑到对于每一个类别的贝叶斯公式的分母均是相同的，因此可以认为$\mathrm{P(Y_k|X)}$和分子成正比，将贝叶斯公式化简为：

$$P(Y_k|X) = \frac{P(Y_K = c_k)}{\sum_i P(X|Y_i)P(Y_i)} \prod_{i=1}^{n} P(X_i = x_i|Y_k) \sim P(Y_K = c_k) \prod_{i=1}^{n} P(X_i = x_i|Y_K = c_k)$$

将朴素贝叶斯用于分类任务可以得到最终的分类类别为：

$$c_{result} = \underset{c_k}{argmax}\, P(Y_K = c_k) \prod_{i=1}^{n} P(X_i = x_i|Y_K = c_k)$$

考虑到 Iris 数据集中的数据均是连续值，认为每一个特征$\mathrm{X_j}$的先验概率均为正态分布，因此可以将上述的$P(X_i = x_i|Y_k = c_k)$具体表示为：

$$P(X_i = x_i|Y_k = c_k) = \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{(X_j-\mu_k)^2}{2\sigma_k^2}}$$

其中μ_k和σ_k是正态分布的均值和标准差，可以通过极大似然估计求得，μ_k是在样本类别c_k中，所有X_j的平均值。σ_k是在样本类别c_k中，所有X_j的标准差，

2. 运行结果

采用和三（一）1.中相同的数据进行计算，即读入数据后将数据集按照 6：4 的比例进行划分，并采用相同的随机数种子 1824。运行朴素贝叶斯代码可以得到在测试集上的准确率为 93.3%小于决策树的 95.56%。可以看到，预测的准确率略低于相同条件下的决策树。[5]

通过运行的结果可以看出，决策树算法相比于朴素贝叶斯具有更高的准确率，但是由于决策树在计算信息熵和信息增益以及选择合适的划分点时需要消耗大量的时间而朴素贝叶斯则不存在这个问题，因此决策树相比于朴素贝叶斯运行速度较慢，特别是在数据量较大时。此外，朴素贝叶斯假设所有的特征都是独立分布的，这与实际情况不一定相符合，因此朴素贝叶斯算法的稳定性不如决策树分类算法。

（二）软决策树（Soft Approach）

在直接运用决策树进行分类时，所有的分类边界都是固定的边界，即是“硬的”。这对于部分处于边界的数据的划分造成了一定的误差。因此可以使用 soft cuts 和 soft DT 方法处理这一部分数据[6]。

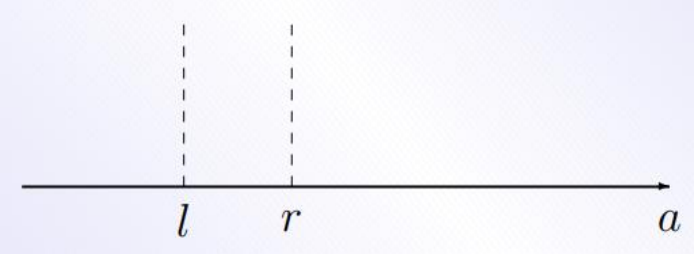


图 6 软分类边界示意图

Soft cuts 的概率由属性值 a，左右边界 l 和 r 组成，$l, r \in R$分别称作概率 p 的左右边界，$\varepsilon = \frac{r-l}{2}$称为概率 p 的不确定半径。当$a(x_1) > r$或者$a(x_2) < l$时，可以直接对数据进行分类，当$a(x) \in [l, r]$时，无法确定$a(x)$是属于左边分支还是右边分支。真实的分类边界

位于[l, r]之间。一般的分类边界可以看作不确定半径为 0.

可以使用 fuzzy decision tree（模糊决策树）和 rough decision tree（粗略决策树）来实现上述的 soft cuts。对于模糊决策树，每一个属于不确定半径内的数据按照一定的概率属于左边分支或者右边分支，再计算该数据到达决策树某一层的概率，该数据最终的标签由概率最大的标签所决定。对于粗略决策树则有，将属于不确定半径内的数据同时分类到树的左边和右边，合并结果，返回结果向量，通过投票得到最优的决策分类。

结论

本文使用决策树的方法对鸢尾花（Iris）数据集进行分类并取得了较为不错的效果，也对决策树的相关性质进行了初步的探究，得到了一些浅显的结论。因为时间关系和数据集较易分类有关，对剪枝算法和 Soft approach 的探究还不够深入，希望之后可以进一步的进行探究。

参考文献

[1]R.A. Fisher, Michael Marshall. Iris Data Set [EB/OL]. http://archive.ics.uci.edu/ml/datasets/Iris,1988-07-01.

[2]Britesun. 决策树算法及 Python 实现[EB/OL]. https://blog.csdn.net/qq_34807908/article/details/81539536,2018-08-09.

[3]空字符.决策树的建模与剪枝[EB/OL].https://zhuanlan.zhihu.com/p/144505469,2020-05-29.

[4]马刚. 朴素贝叶斯算法的改进与应用[D].安徽大学,2018.

[5]朱比特.朴素贝叶斯 python 复现（基于数据集 iris）[EB/OL]. https:// blog.csdn.net/weixin_52328678/article/details/115628248,2021-04-12.

[6] Nguyen, Hung Son. (2002). A Soft Decision Tree. 57-66. 10.1007/978-3-7908-1777-5_6.