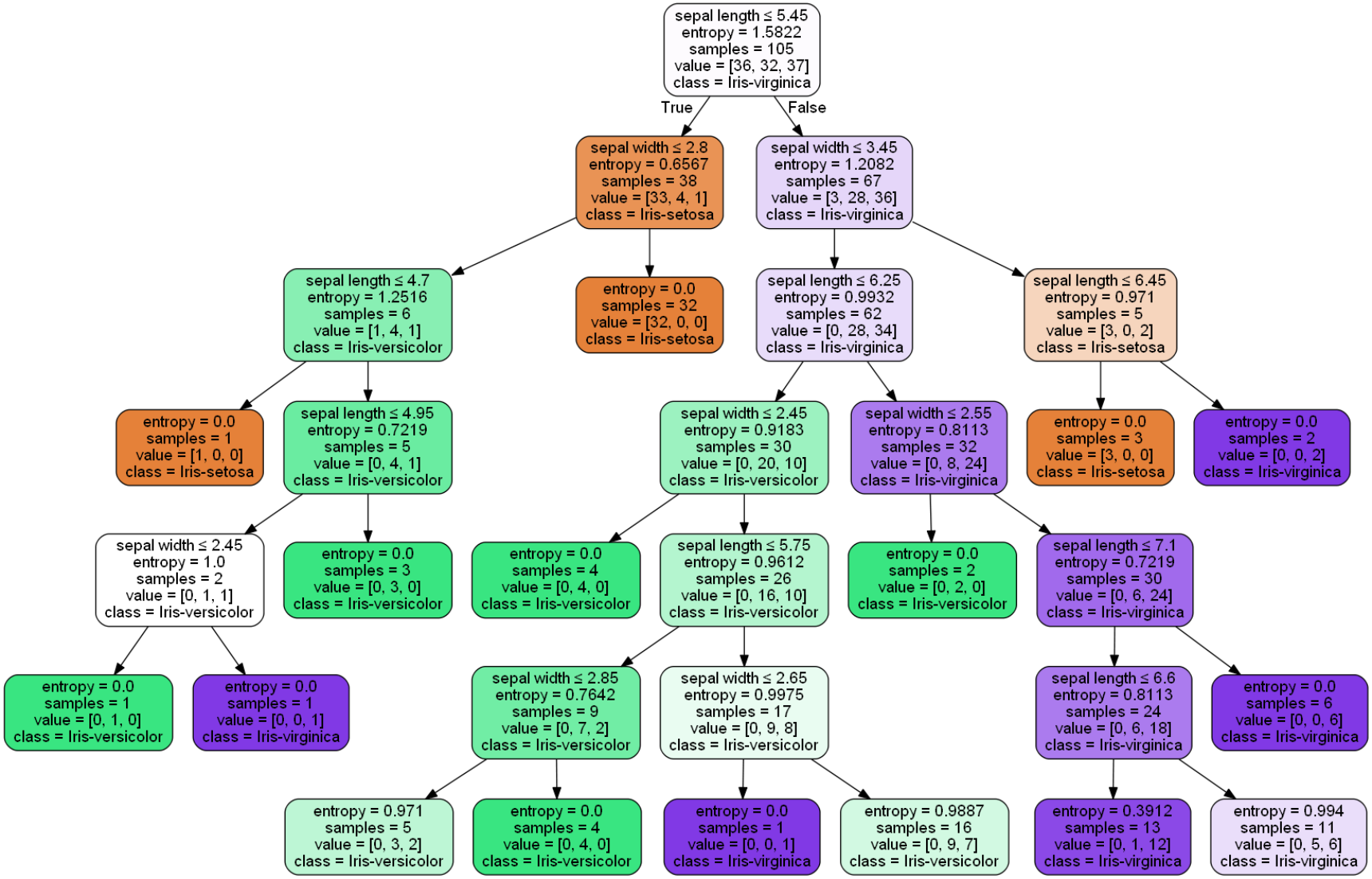
决策树&Bagging方法

王梓鉴

目录

概述


sepal length ≤ 5.45
entropy = 1.5822
samples = 105
value = [36, 32, 37]
class = Iris-virginica
True
False
sepal width ≤ 2.8
entropy = 0.6567
samples = 38
value = [33, 4, 1]
class = Iris-setosa
sepal width ≤ 3.45
entropy = 1.2082
samples = 67
value = [3, 28, 36]
class = Iris-virginica
sepal length ≤ 4.7
entropy = 1.2516
samples = 6
value = [1, 4, 1]
class = Iris-versicolor
entropy = 0.0
samples = 32
value = [32, 0, 0]
class = Iris-setosa
sepal length ≤ 6.25
entropy = 0.9932
samples = 62
value = [0, 28, 34]
class = Iris-virginica
sepal length ≤ 6.45
entropy = 0.971
samples = 5
value = [3, 0, 2]
class = Iris-setosa
entropy = 0.0
samples = 1
value = [1, 0, 0]
class = Iris-setosa
sepal length ≤ 4.95
entropy = 0.7219
samples = 5
value = [0, 4, 1]
class = Iris-versicolor
sepal width ≤ 2.45
entropy = 0.9183
samples = 30
value = [0, 20, 10]
class = Iris-versicolor
sepal width ≤ 2.55
entropy = 0.8113
samples = 32
value = [0, 8, 24]
class = Iris-virginica
entropy = 0.0
samples = 3
value = [3, 0, 0]
class = Iris-setosa
entropy = 0.0
samples = 2
value = [0, 0, 2]
class = Iris-virginica
sepal width ≤ 2.45
entropy = 1.0
samples = 2
value = [0, 1, 1]
class = Iris-versicolor
entropy = 0.0
samples = 3
value = [0, 3, 0]
class = Iris-versicolor
entropy = 0.0
samples = 4
value = [0, 4, 0]
class = Iris-versicolor
sepal length ≤ 5.75
entropy = 0.9612
samples = 26
value = [0, 16, 10]
class = Iris-versicolor
entropy = 0.0
samples = 2
value = [0, 2, 0]
class = Iris-versicolor
sepal length ≤ 7.1
entropy = 0.7219
samples = 30
value = [0, 6, 24]
class = Iris-virginica
entropy = 0.0
samples = 1
value = [0, 1, 0]
class = Iris-versicolor
entropy = 0.0
samples = 1
value = [0, 0, 1]
class = Iris-virginica
sepal width ≤ 2.85
entropy = 0.7642
samples = 9
value = [0, 7, 2]
class = Iris-versicolor
sepal width ≤ 2.65
entropy = 0.9975
samples = 17
value = [0, 9, 8]
class = Iris-versicolor
sepal length ≤ 6.6
entropy = 0.8113
samples = 24
value = [0, 6, 18]
class = Iris-virginica
entropy = 0.0
samples = 6
value = [0, 0, 6]
class = Iris-virginica
entropy = 0.971
samples = 5
value = [0, 3, 2]
class = Iris-versicolor
entropy = 0.0
samples = 4
value = [0, 4, 0]
class = Iris-versicolor
entropy = 0.0
samples = 1
value = [0, 0, 1]
class = Iris-virginica
entropy = 0.9887
samples = 16
value = [0, 9, 7]
class = Iris-versicolor
entropy = 0.3912
samples = 13
value = [0, 1, 12]
class = Iris-virginica
entropy = 0.994
samples = 11
value = [0, 5, 6]
class = Iris-virginica

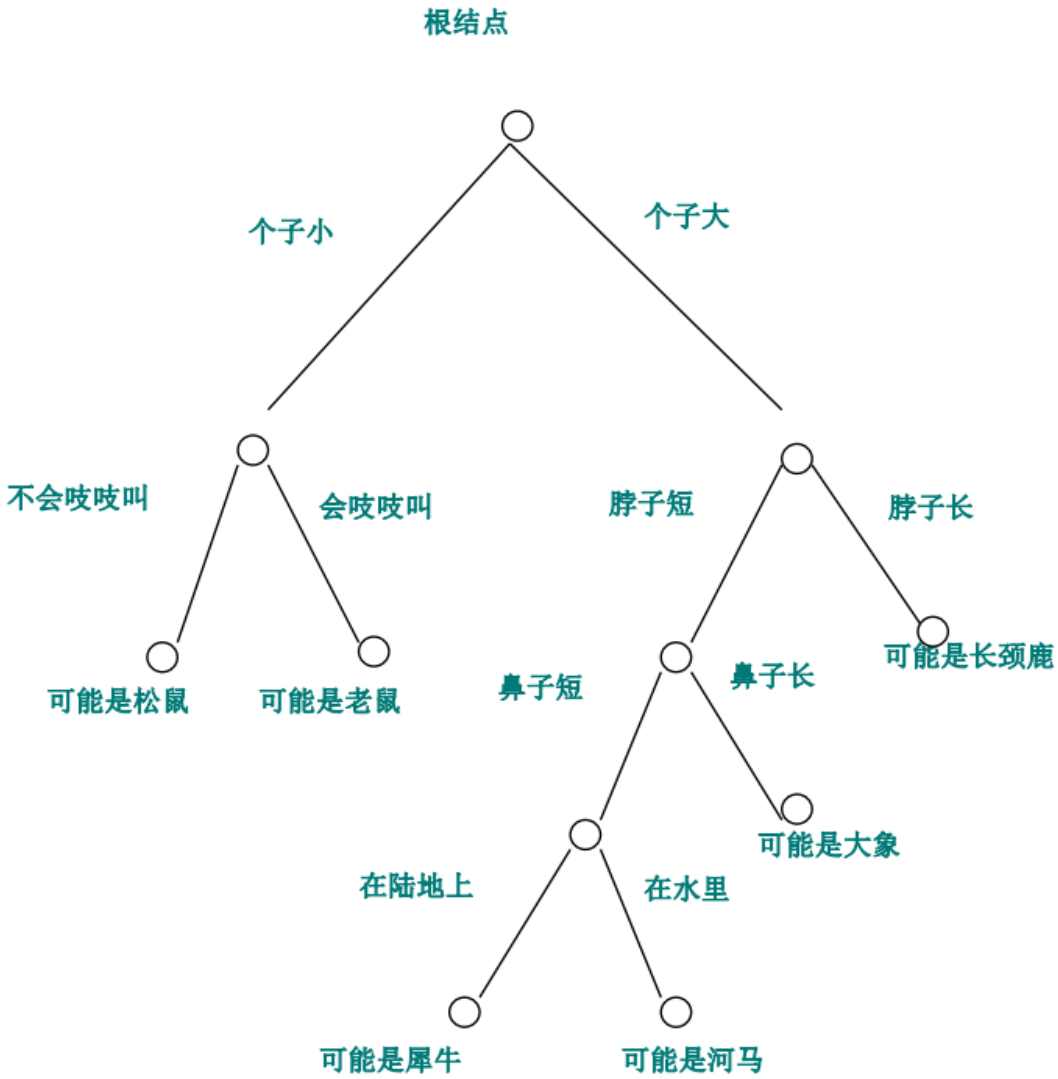
概述

- 用于分类、回归
- If-then规则（if条件then结论）的集合/对特征空间的划分
- 一般3个步骤：特征选择、决策树的生成、剪枝处理
- 决策树的组成：节点、有向边
- 节点：（一个）根节点、内部节点、叶节点
- 根节点：样本全集
- 内部节点：表示一个特征，继续向下划分
- 叶节点：表示一类，不再向下划分
- 选取最优决策树是NP完全问题

（无法确定是否能在多项式时间内求解，但可以用多项式时间验证）

- 启发式方法（相对于最优化算法——基于直观和经验求局部最优解）
- 解决过拟合问题：剪枝处理

概述


根结点
个子小
个子大
不会吱吱叫
会吱吱叫
脖子短
脖子长
可能是松鼠
可能是老鼠
鼻子短
鼻子长
可能是长颈鹿
可能是大象
在陆地上
在水里
可能是犀牛
可能是河马

概述

- 决策树基本思路

输入： 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
　　属性集 $A = \{a_1, a_2, \ldots, a_d\}$.

过程： 函数 TreeGenerate(D, A)

```
1: 生成结点 node;
2: if D 中样本全属于同一类别 C then
3:     将 node 标记为 C 类叶结点; return
4: end if
5: if A = ∅ OR D 中样本在 A 上取值相同 then
6:     将 node 标记为叶结点, 其类别标记为 D 中样本数最多的类; return
7: end if
8: 从 A 中选择最优划分属性 a_*;
9: for a_* 的每一个值 a_*^v do
10:    为 node 生成一个分支; 令 D_v 表示 D 中在 a_* 上取值为 a_*^v 的样本子集;
11:    if D_v 为空 then
12:        将分支结点标记为叶结点, 其类别标记为 D 中样本最多的类; return
13:    else
14:        以 TreeGenerate(D_v, A \ {a_*})为分支结点
15:    end if
16: end for
```

输出： 以 node 为根结点的一棵决策树

图 4.2 决策树学习基本算法

概述

决策树的生成是个递归的过程，显然能发现三种导致递归返回的情况：

1、当前节点所包含的样本全部属于同一类，无需划分 。这时将结点化为叶子结点，样本属于该类别。

2、属性集为空或者数据集在当前属性集上所有取值相同，无法划分 。这时将结点化为叶子结点并将样本归属于多数类。

3、当前节点所包含的样本集合为空，不能划分。这时将结点化为叶子结点并将样本归属于**父节点的**多数类。

划分原则?

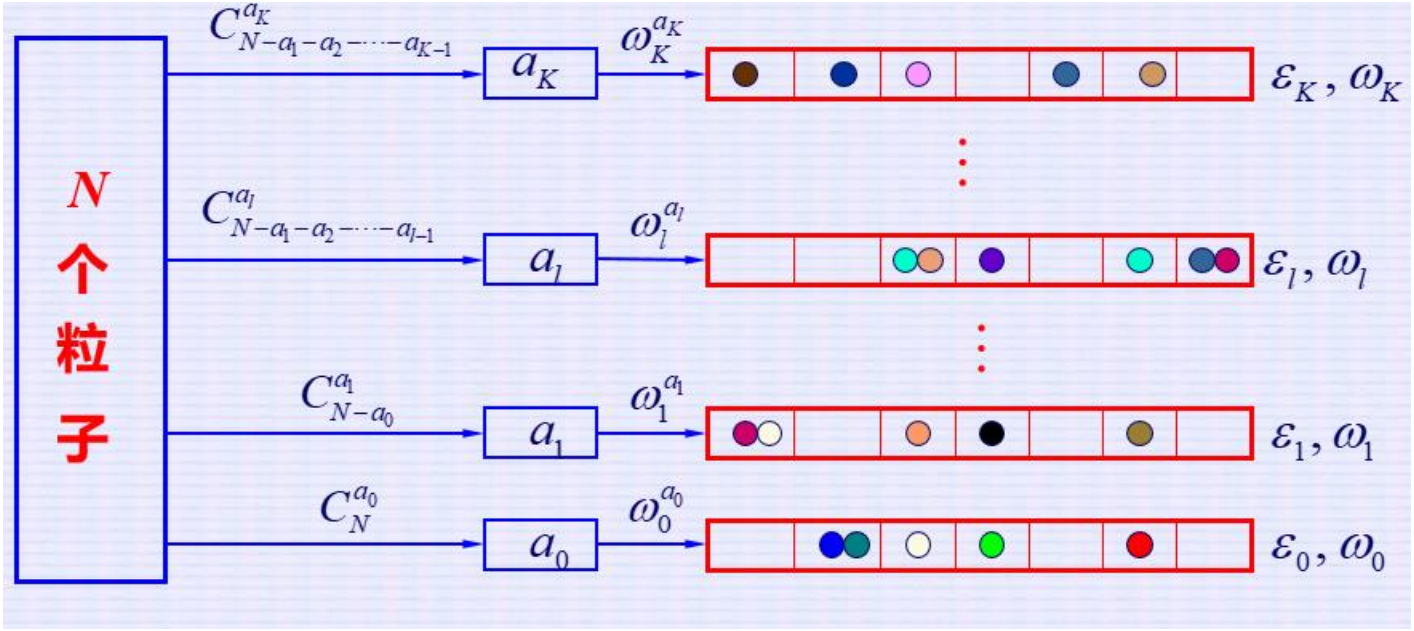
信息熵

- 决策树划分目标：使分支结点所包含的样本尽可能属于同一类别
- 熵：混沌/不确定程度的数学度量
  - 热力学上，玻尔兹曼熵是用于表征系统的不确定程度的物理量

$$S = k\ln\Omega$$

k：玻尔兹曼常数

Ω：系统的微观态个数



信息熵

- 若用 $H(p_1, p_2, \dots, p_n)$ 记样本空间 $(p_1, p_2, \dots, p_n)$所对应的不确定度，运用同样的直觉分析，我们相信当所有的基本事件机会均等，即都有同样的概率1/n时，其不确定度最大。因而，不确定度函数H应该满足如下的基本不等式：对所有的加起来等于1的非负“概率数” $p_1, p_2, \dots, p_n$，有：

$$H(p_1, p_2, \dots, p_n) \leq H\left(\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n}\right)$$

- 投掷骰子，投掷的次数越多，直观上总的投掷结果不确定度越大，将这种直观一般化，我们就有不确定度函数H 应该满足的单调性要求：

$H\left(\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n}\right)$是n的严格单调递增函数（单调性）

- 极端情况：完全确定的时间不包含任何信息量

信息熵

- 假设物理系赵教授、数学系钱教授和孙教授竞争理学院的一笔科研基金，他们每人申请成功的概率分别为$1/2, 1/3, 1/6$。院长为求公平，让每个系得此奖励的机会均等。若物理系拿到资助，就到了赵教授的名下。如数学系得到了它，钱教授有2/3的概率拿到，孙教授则有1/3的机会到手。通过分析“条件概率”，我们能得出不确定度 $H(1/2, 1/3, 1/6)$ 的数值：这三个教授获得基金的不确定度，等于物理系或数学系拿到这笔基金的不确定度，加上数学系赢得该基金的概率与在数学系拿到基金的条件之下，钱教授或孙教授得到它的不确定度之乘积。换言之，$H(1/2, 1/3, 1/6) = H(1/2, 1/2) + \frac{1}{2}H(2/3, 1/3)$。推而广之，可以得出不确定度与条件概率有关的“**加权和**”性质：**多随机事件同时发生存在的总不确定性的量度是可以表示为各事件不确定性的量度的和（累加性）**

信息熵

- **不确定函数是非负的。**显然，由累加性，增加了某一事件，总不确定度一定增加，故不确定函数非负。
- 综合以上的递增性、累加性、非负性，自然有以下表达式：

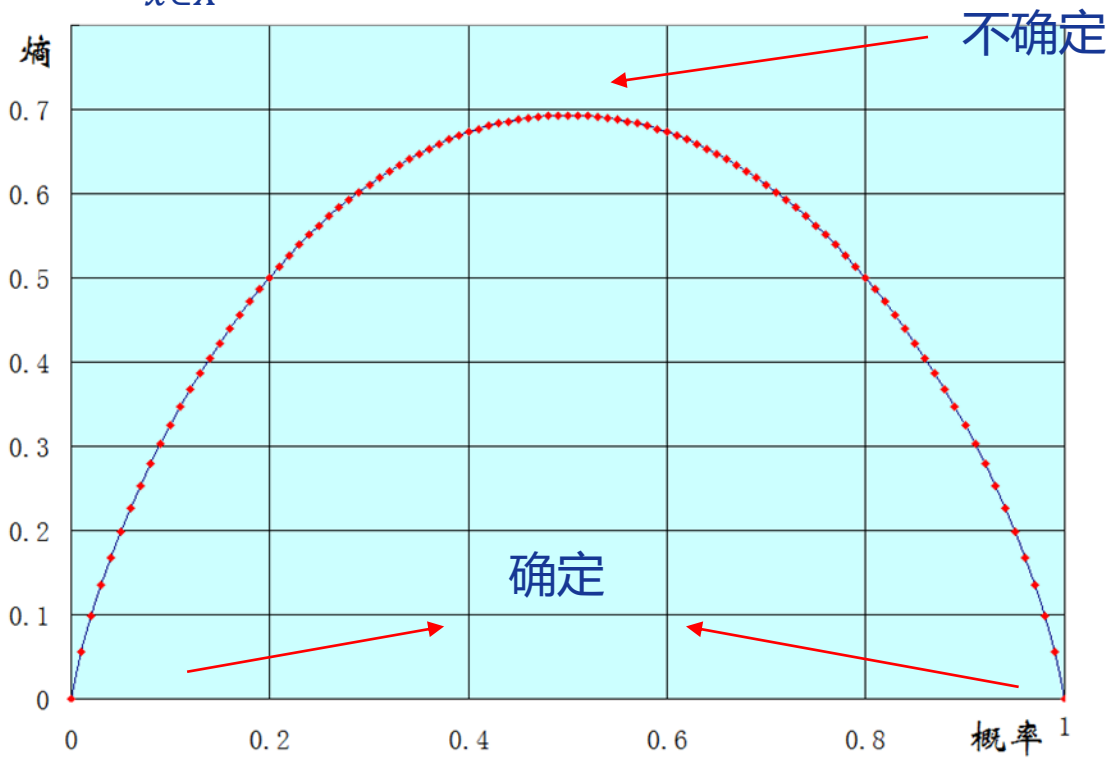
$$H(p_1, p_2, \dots, p_n) = \sum_{\mathrm{i}}^{n} -p_i log p_i$$

- 以上H函数被定义为信息熵
- 信息熵的单位根据上式对数的底数不同而有不同的单位：bit（底数为2），nat（底数为e），Hart（底数为10）

信息熵

- 以二项分布为例

$$H(X) = -\sum_{x \in X} p(x) \ln p\,(x) = -p \ln p - (1-p) \ln(1-p)$$



特征选择

- 条件熵

$$H(Y|X) = \sum_{i}^{n} P(X = x_i) H(Y|X = x_i)$$

- 信息增益（用特征a对样本D进行划分得到D^v）

$$Gain(D, a) = H(D) - H(Y|a),$$

$$或 g(D, a) = H(D) - \sum_{v} \frac{|D^v|}{|D|} H(D^v)$$

其中，$|D|$为样本D的样本数

- 信息增益表示得知特征a的信息，使得样本D分类的确定性（数据的“纯度”）提升程度。显然，选取分类特征时，信息增益越大越好
- 在决策树模型中，信息增益称为互信息

特征选择

- 信息增益率（用特征a对样本D进行划分得到D^v）

$$Gain\ ratio(D,a) = \frac{Gain(D,a)}{IV(a)}$$

其中，$IV(a) = -\sum_v \frac{|D^v|}{|D|} log_2 \frac{|D^v|}{|D|}$，只与a有关，称为a的“固有值”

- 与信息增益**率**对可能取值数目少的属性有所偏好；信息增益对可能取值数目多的属性有所偏好
- 基尼值（Gini）

假设数据集有γ类，样本属于第k类的概率为p_k，则样本的的基尼值为

$$Gini(D) = 1 - \sum_k^{\gamma} p_k^2$$

- Gini(D)越**小**，数据集的纯度越高

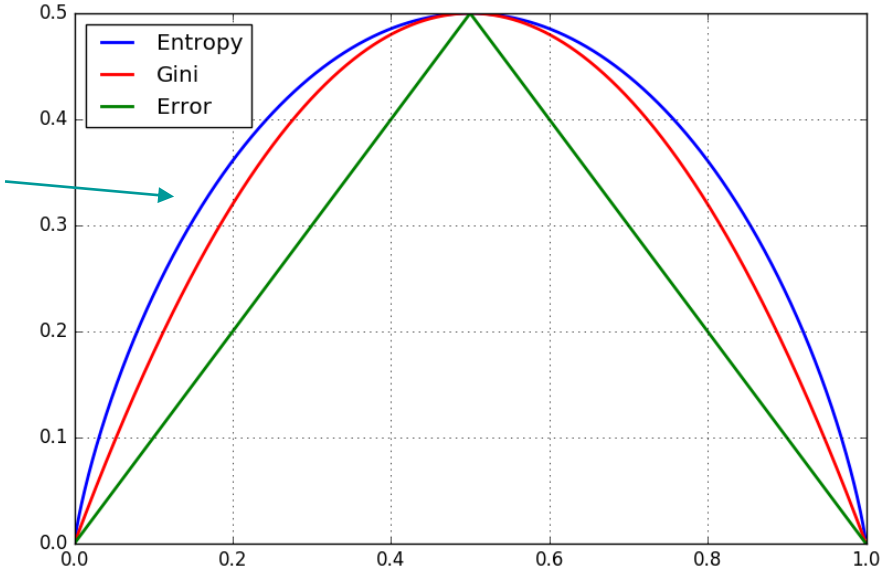
特征选择

- （特征a的）基尼指数（Gini index），即数据集D对特征a进行划分后的基尼指数

$$Gini\ index(D,a) = \sum_{v} \frac{|D^v|}{|D|} Gini(\mathrm{D}^v)$$

- 三种指标比较（二分类）

半熵



决策树的生成

算法	划分指标
ID3	信息增益
C4.5	信息增益率
CART	基尼指数

决策树的生成

- 输入：训练数据集D，特征集A，阈值ε；
- 输出：决策树T。

1. 若D中所有实例属于同一类C_k，则T为单结点树，并将类C_k作为该结点的类标记，返回T；
2. 若A = Ø，则T为单结点树，并将D中实例数最大的类C_k作为该结点的类标记，返回T；
3. 否则，计算A中各特征对D的信息增益/信息增益率（基尼指数），选择信息增益/信息增益率**最大**（基尼指数最小）的特征A_g；
4. 如果A_g的信息增益/信息增益率/基尼指数小于阈值ε，则置T为单结点树，并将D中实例数最大的类C_k作为该结点的类标记，返回T；
5. 否则，对A_g的每一可能值a_i，依$A_g = a_i$将D分割为若干非空子集D_i，将D_i中实例数最大的类作为标记，构建子结点，由结点及其子结点构成树T，返回T；
6. 对第i个子结点，以D_i为训练集，以A-{A_g}为特征集，递归地调用步1~步5得到子树T_i，返回T_i。

剪枝处理

- 解决过拟合问题
- 例如，以数据集序号为特征对数据集进行划分，信息增益（ID3）最大，但得到的结果不具有泛化能力
- 预剪枝：在决策树生成的过程中，在每个结点划分时进行估计，如果，该划分不能提高决策树的泛化能力，则停止划分，以当前结点为叶结点（以最大比例的类为整体的类）
- 后剪枝：生成完整的决策树后自下而上地向根结点进行考察，若将一非叶结点替换为叶结点能提高决策树的泛化能力，则进行该替换
- 预剪枝的缺点：易导致欠拟合

一种剪枝方法

- 决策树整体的损失函数

设树T的叶结点个数为|T|, t是树T的叶结点，该叶结点有N_t个样本点，其中k类的样本点有N_{tk}个，k=1,2,...,K， $H_t(T)$为叶结点t上的经验熵（条件熵），$a \geq 0$为参数，损失函数为

$$C_\alpha(T) = \sum_{t}^{|T|} N_t H_{t(T)} + \alpha|T|$$

上式中，第一项表示模型在训练集上的预测误差，通常记作C(T)，C(T)也可以为基尼指数，平方误差等；第二项表示模型的复杂度。α越大，表示更倾向于选择结构更简单的决策树；α=0代表在损失函数中不考虑模型的复杂度。

- 剪枝思路：选择损失函数最小的模型（子树）

一种剪枝方法

- 输入：生成的完整决策树，参数α
- 输出：剪枝后的子树

1. 计算每个结点的条件熵
2. 递归地从树的叶结点向上回缩，若回缩后损失函数减小，则进行剪枝，即父结点变为子结点
3. 反复执行步2直到不能继续为止，得到损失函数最小的子树。

CART剪枝

- CART剪枝算法由两步组成首先从生成算法产生的决策树T_0底端开始不断剪枝，直到T_0的根结点，形成一个子树序列$\{T_0, T_1, \cdots, T_n\}$；然后通过交叉验证法在独立的验证数据集上对子树序列进行测试，从中选择最优子树。
- Step1（确定α）

用递归的方法对树进行剪枝，将α从小增大，$\alpha_0 < \alpha_1 < \alpha_2 < \cdots \alpha_n < +\infty$，产生一系列的区间$[\alpha_1, \alpha_{i+1})$，$i = 0,1, \ldots, n$；剪枝得到的子树序列对应着区间$[\alpha_1, \alpha_{i+1})$，$i = 0,1, \ldots, n$的最优子树序列$\{T_0, T_1, \cdots, T_n\}$，序列中的子树是嵌套的。

对T_0中每一内部结点t，计算$g(t) = \frac{C(t) - C(T_t)}{|T_t| - 1}$，表示剪枝后整体损失函数减少的程度，在$T_0$中剪去$g(t)$最小的$T_t$，将得到的子树作为$T_1$，同时将最小的$g(t)$设为a1，$T_1$为区间$[\alpha_1, \alpha_2)$的最优子树。如此剪枝下去，直至得到根结点。在这一过程中，不断地增加α的值，产生新的区间。

CART剪枝

- Step2（逐级判断是否缩回）

在剪枝得到的子树序列$\{T_0, T_1, \cdots, T_n\}$中通过交叉验证选取最优子树T。

利用独立的验证数据集，测试子树序列$\{T_0, T_1, \cdots, T_n\}$中各棵子树的平方误差或基尼指数。平方误差或基尼指数最小的决策树被认为是最优的决策树。在子树序列中，每棵子树$\{T_0, T_1, \cdots, T_n\}$都对应于一个参数$\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_n$。所以，当最优子树$T_k$确定时，对应的$\alpha_k$也确定了，即得到最优决策树$T_\alpha$。

回归树

- 一个回归树对应着输入空间（即特征空间）的一个划分以及在划分单元上的输出值.与分类树不同的是，回归树对输入空间的划分采用一种启发式的方法，会遍历所有输入变量，找到最优的切分变量j和最优的切分点s，即选择第j个特征$x^{(j)}$和它的取值s将输入空间划分为两部分，然后重复这个操作。而如何找到最优的j和s是通过比较不同的划分的误差来得到的。一个输入空间的划分的误差是用真实值和划分区域的预测值的最小二乘来衡量的，即

$$\sum_{x \in R_m} \left(y_i - f_{(x_i)}\right)^2$$

回归树算法

1. 求解最优切分变量j和切分点s。

对全部数据进行遍历，对每一个j搜索对应的s，满足：

$$\min_{j,s}[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i-c_2)^2]$$

2. 用选定的对(j,s)划分区域并决定相应的输出值

$$R_1(j,s)=\{x|x^{(j)}\le s\},R_2(j,s)=\{x|x^{(j)}>s\}$$

$$\hat{c}_m=\frac{1}{N_m}\sum_{x_i\in R_m(j,s)}y_i,x_i\in R_m,m=1,2$$

3. 继续对两个子区域调用步1、2，直到满足停止条件
4. 将输入空间划分为M个区域$R_1,R_2,\dots R_m$，生成决策树

$$f(x)=\sum_{m}^{M}\hat{c}_m I(\mathrm{x}\in R_m)$$

回归树

臂长（m）	年龄(岁)	体重（kg）	身高（m）(标签值)
0.5	5	20	1.1
0.7	7	30	1.3
0.9	21	70	1.7

以第一个特征的第一个值为划分点（0.5），则划分的两个空间记为R_1, R_2

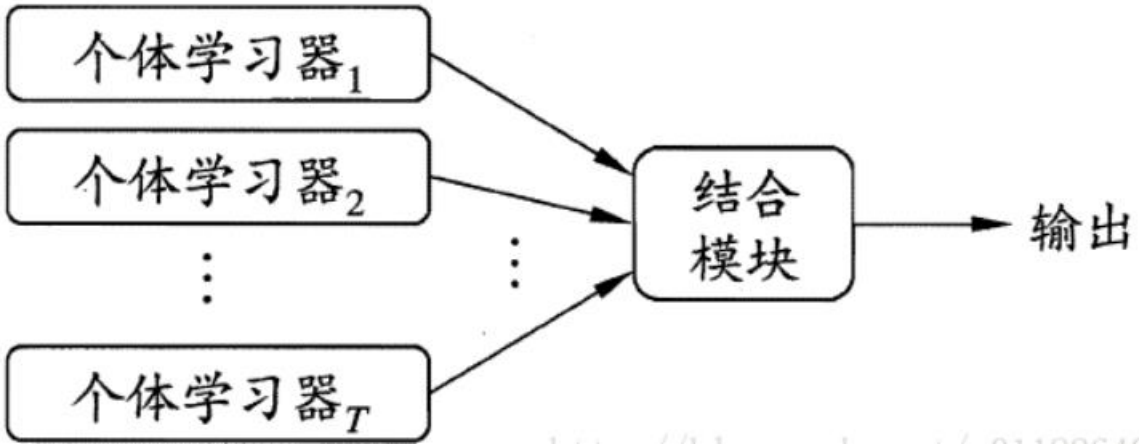
R_1={(0.5,5,20)}, R_2 = {(0.7,7,30),(0.9,21,70)}

$$c_1 = 1.1, c_2 = \frac{1.3 + 1.7}{2} = 1.5$$

所以平方误差$m(0.5) = (1.1 - 1.1)^2 + (1.5 - 1.3)^2 + (1.5 - 1.7)^2 = 0.08$

Bagging

- 集成学习（ensemble learning）指的是将多个学习器进行有效地结合，共同完成学习任务



- 个体学习器可以同属同一类别（同质集成），也可以不同属（异质集成）
- 目标：每个个体学习器尽可能独立

Bagging

- Bagging是一种并行式集成学习方法（个体学习器之间没有前后顺序）
- 使用“有放回”采样的方式选取训练集，对于包含m个样本的训练集，进行m次有放回的随机采样操作，从而得到m个样本的采样集，这样训练集中有接近36.8%的样本没有被采到。按照相同的方式重复进行，我们就可以采集到T个包含m个样本的数据集，从而训练出T个基学习器，最终对这T个基学习器的输出进行结合。

$$\lim_{m \mapsto \infty} \left(1 - \frac{1}{m}\right)^m \mapsto \frac{1}{e} \approx 0.368$$

- 结合方法：对分类任务使用简单投票法，回归任务采用简单平均法
- Bagging的优点

1. 高效：集成后的复杂度与个体学习器的复杂度同阶：
2. Bagging只使用了63.2%左右的数据样本，剩余的可用作验证集

调个包

In [1]:
```
from sklearn.datasets import load_iris
from sklearn import tree
from sklearn.tree import export_graphviz
from IPython.display import Image
import pydotplus
```

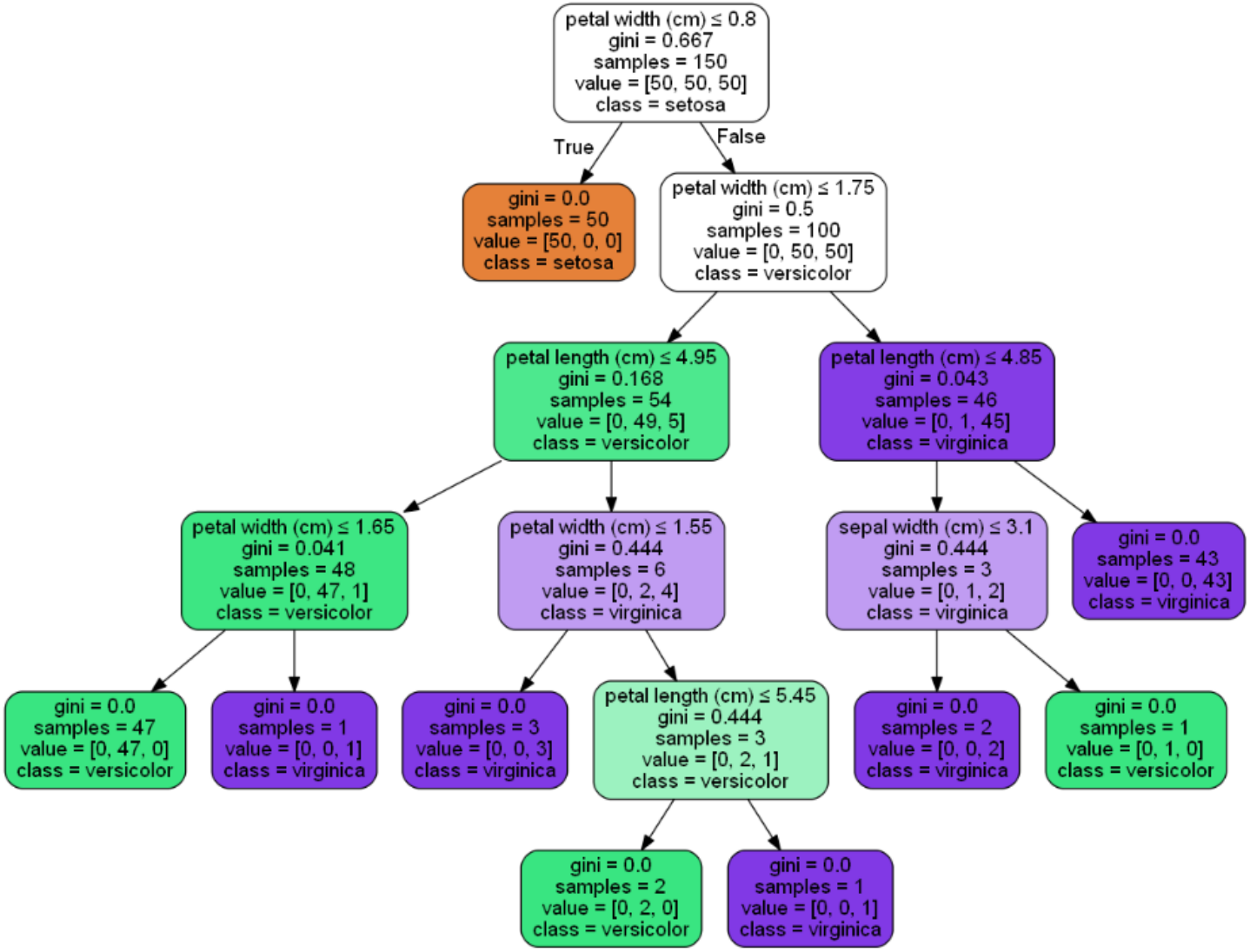
In [2]:
```
iris = load_iris()
tree = tree.DecisionTreeClassifier(random_state=0)
tree.fit(iris.data, iris.target)
```

Out[2]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=0,
            splitter='best')
```

In [3]:
```
import os
os.environ["PATH"] += os.pathsep + 'C:/Program Files (x86)/Graphviz2.38/bin/'
```

In [4]:
```
dot_data = export_graphviz(tree, out_file=None,
                           feature_names=iris.feature_names,
                           class_names=iris.target_names,
                           filled=True, rounded=True,
                           special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

调个包


petal width (cm) ≤ 0.8
gini = 0.667
samples = 150
value = [50, 50, 50]
class = setosa
True
False
gini = 0.0
samples = 50
value = [50, 0, 0]
class = setosa
petal width (cm) ≤ 1.75
gini = 0.5
samples = 100
value = [0, 50, 50]
class = versicolor
petal length (cm) ≤ 4.95
gini = 0.168
samples = 54
value = [0, 49, 5]
class = versicolor
petal length (cm) ≤ 4.85
gini = 0.043
samples = 46
value = [0, 1, 45]
class = virginica
petal width (cm) ≤ 1.65
gini = 0.041
samples = 48
value = [0, 47, 1]
class = versicolor
petal width (cm) ≤ 1.55
gini = 0.444
samples = 6
value = [0, 2, 4]
class = virginica
sepal width (cm) ≤ 3.1
gini = 0.444
samples = 3
value = [0, 1, 2]
class = virginica
gini = 0.0
samples = 43
value = [0, 0, 43]
class = virginica
gini = 0.0
samples = 47
value = [0, 47, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica
gini = 0.0
samples = 3
value = [0, 0, 3]
class = virginica
petal length (cm) ≤ 5.45
gini = 0.444
samples = 3
value = [0, 2, 1]
class = versicolor
gini = 0.0
samples = 2
value = [0, 0, 2]
class = virginica
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
gini = 0.0
samples = 2
value = [0, 2, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica

Reference

- 李航《统计学习方法》（第2版）
- 周志华《机器学习》
- https://www.zhihu.com/question/22178202
- https://zhuanlan.zhihu.com/p/36108972
- https://blog.csdn.net/aaa_aaa1sdf/article/details/81588382
- https://blog.csdn.net/sun_xiao_kai/article/details/88948356