# 决策树算法

“决策树的生成只考虑局部最优，
决策树的剪枝则考虑全局最优。”

七月在线

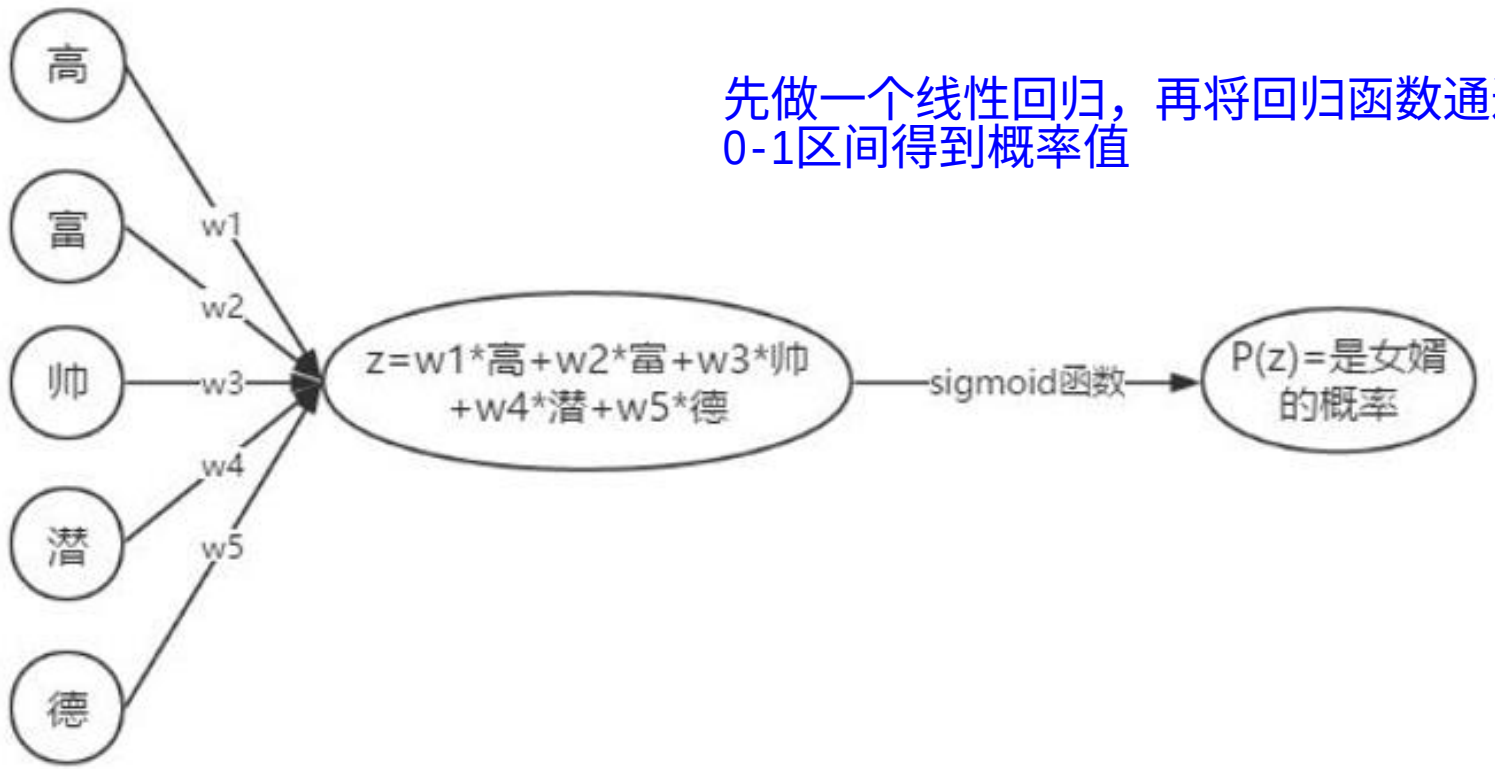
# 主要内容

- 从LR到决策树
  - 1. 总体流程与核心问题
  - 2. 熵、信息增益、信息增益率
- 回归树
  - 1.构建回归树
  - 2.最优化回归树
- 从决策树到随机森林
  - 1.采样与bootstrap
  - 2.bagging与随机森林

# 从LR到决策树

- 思考一下一个分类问题：是否去相亲

LR的解决办法可能是这样的

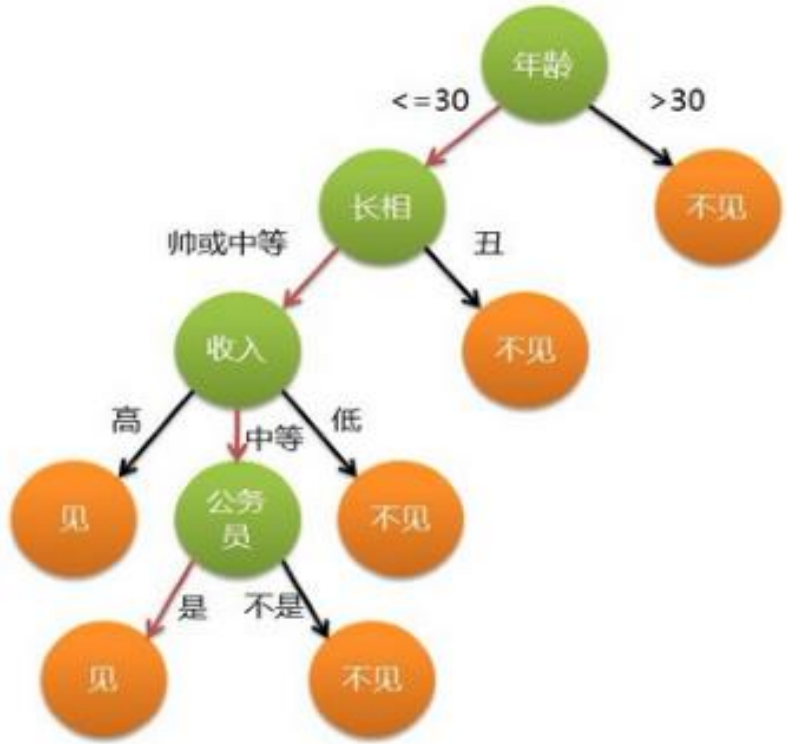



先做一个线性回归，再将回归函数通过sigmoid函数映射到0-1区间得到概率值

# 从LR到决策树

- 思考一下一个分类问题：是否去相亲

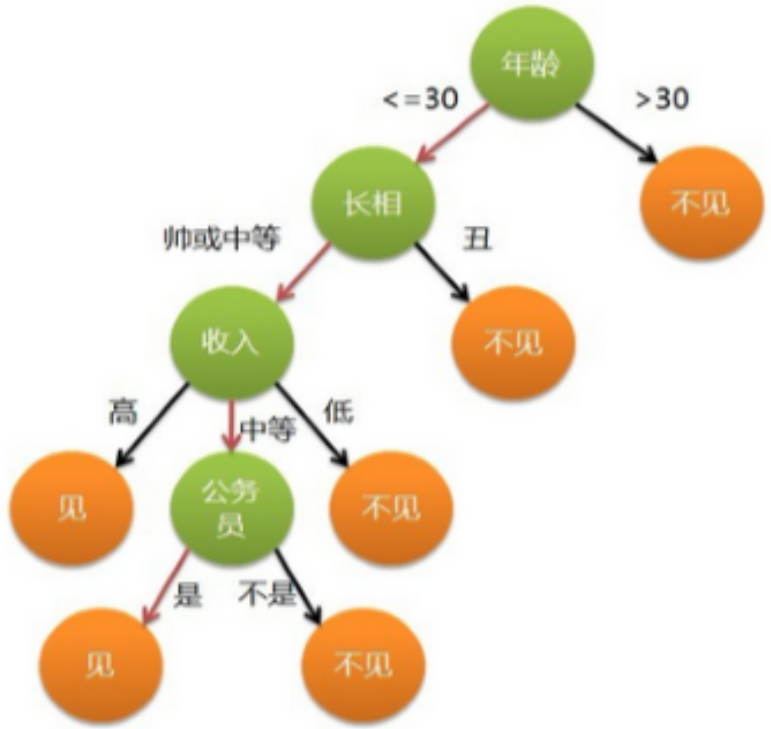
可是有时候，人更直观的方式是这样的



# 从LR到决策树

决策树模型(Decision Tree model) 是一个模拟人类决策过程思想的模型，以找对象为例，一个女孩的母亲要给这个女孩介绍男朋友，于是有了下面的对话：

女儿：多大年纪了？ (年龄)
母亲：26
女儿：长的帅不帅？ (长相)
母亲：挺帅的
女儿：收入高不？ (收入情况)
母亲：不算很高，中等情况
女儿：是公务员不？ (是否公务员)
母亲：是，在税务局上班呢。
女儿：那好，我去见见

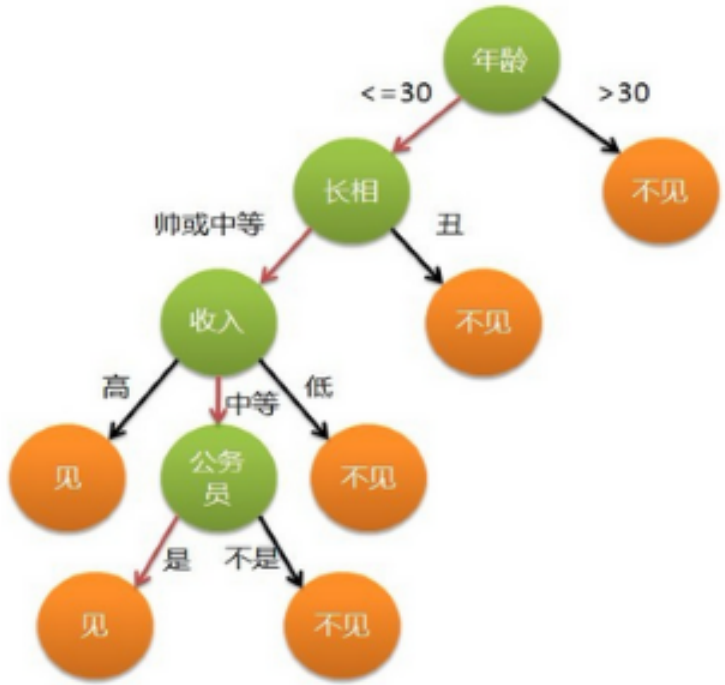



简单、逻辑清晰、可解释性好

# 从LR到决策树

决策树基于“树”结构进行决策

- 每个“内部结点”对应于某个属性上的“测试”
- 每个分支对应于该测试的一种可能结果（即该属性的某个取值）
- 每个“叶结点”对应于一个“预测结果”




学习过程：通过对训练样本的分析来确定“划分属性”（即内部结点所对应的属性）

预测过程：将测试示例从根结点开始，沿着划分属性所构成的“判定测试序列”下行，直到叶结点

# 决策树总体流程

总体流程：

“分而治之”(divide-and-conquer)

- 自根至叶的递归过程
- 在每个中间结点寻找一个“划分”(split or test)属性

三种停止条件：

- 当前结点包含的样本全属于同一类别，无需划分；
- 当前属性集为空, 或是所有样本在所有属性上取值相同，无法划分；
- 当前结点包含的样本集合为空，不能划分。

# 决策树总体流程

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**

前面的(1)情形
递归返回

5: **if** $A = \varnothing$ **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**

前面的(2)情形
递归返回

利用当前结点的后验分布

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($D_v$, $A \setminus \{a_*\}$)为分支结点
15: 　**end if**
16: **end for**

前面的(3)情形
递归返回

将父结点的样本分布作为
当前结点的先验分布

决策树算法的核心

输出：以 node 为根结点的一棵决策树

# 核心数学概念：熵

信息熵 (entropy) 是度量样本集合“纯度”最常用的一种指标，假定当前样本集合 D 中第 k 类样本所占的比例为$p_k$，则 D 的信息熵定义为：

和逻辑回归的损失函数相同

$$\mathrm{Ent}(D) = -\sum_{k=1}^{|\mathcal{Y}|} p_k \log_2 p_k$$

计算信息熵时约定：若 $p = 0$，则 $p \log_2 p = 0$.

需要计算熵的，叶子节点一定不满足三条停止准则

$\mathrm{Ent}(D)$的值越小，则$D$的纯度越高

$\mathrm{Ent}(D)$ 的最小值为 0，最大值为 $\log_2 |\mathcal{Y}|$.

信息增益直接以信息熵为基础，计算当前划分对信息熵所造成的变化。

# 最佳划分属性选择：信息增益

信息增益 (information gain)：ID3中使用

离散属性 $a$ 的取值 $\{a^1, a^2, a^3, \dots a^V\}$：

$D^v$：$D$ 中在 $a$ 上取值 = $a^v$ 的样本集合

以属性 $a$ 对 数据集 $D$ 进行划分所获得的信息增益为：

$$Gain(D,a) = Ent(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|} Ent(D^v)$$

划分前的信息熵　划分后的信息熵

第v个分支的权重，样本越多越重要

当前叶子节点的人数占比作为该节点的权重。

# 最佳划分属性选择：信息增益

信息增益示例：

该数据集包含17个 训练样例，结果有2个类别$|y| = 2$，其中正例占$P_1 = \frac{8}{17}$ 反例占$P_2 = \frac{9}{17}$

根结点的信息熵为

$$Ent(D) = -\sum_{k=1}^{2} p_k log_2^{p_k}$$

$$= -\left(\frac{8}{17} log_2^{\frac{8}{17}} + \frac{9}{17} log_2^{\frac{9}{17}}\right) = 0.998$$

周志华老师《机器学习》西瓜数据集

| 编号 | 色泽 | 根蒂 | 敲声 | 纹理 | 脐部 | 触感 | 好瓜 |
|---|---|---|---|---|---|---|---|
| 1 | 青绿 | 蜷缩 | 浊响 | 清晰 | 凹陷 | 硬滑 | 是 |
| 2 | 乌黑 | 蜷缩 | 沉闷 | 清晰 | 凹陷 | 硬滑 | 是 |
| 3 | 乌黑 | 蜷缩 | 浊响 | 清晰 | 凹陷 | 硬滑 | 是 |
| 4 | 青绿 | 蜷缩 | 沉闷 | 清晰 | 凹陷 | 硬滑 | 是 |
| 5 | 浅白 | 蜷缩 | 浊响 | 清晰 | 凹陷 | 硬滑 | 是 |
| 6 | 青绿 | 稍蜷 | 浊响 | 清晰 | 稍凹 | 软粘 | 是 |
| 7 | 乌黑 | 稍蜷 | 浊响 | 稍糊 | 稍凹 | 软粘 | 是 |
| 8 | 乌黑 | 稍蜷 | 浊响 | 清晰 | 稍凹 | 硬滑 | 是 |
| 9 | 乌黑 | 稍蜷 | 沉闷 | 稍糊 | 稍凹 | 硬滑 | 否 |
| 10 | 青绿 | 硬挺 | 清脆 | 清晰 | 平坦 | 软粘 | 否 |
| 11 | 浅白 | 硬挺 | 清脆 | 模糊 | 平坦 | 硬滑 | 否 |
| 12 | 浅白 | 蜷缩 | 浊响 | 模糊 | 平坦 | 软粘 | 否 |
| 13 | 青绿 | 稍蜷 | 浊响 | 稍糊 | 凹陷 | 硬滑 | 否 |
| 14 | 浅白 | 稍蜷 | 沉闷 | 稍糊 | 凹陷 | 硬滑 | 否 |
| 15 | 乌黑 | 稍蜷 | 浊响 | 清晰 | 稍凹 | 软粘 | 否 |
| 16 | 浅白 | 蜷缩 | 浊响 | 模糊 | 平坦 | 硬滑 | 否 |
| 17 | 青绿 | 蜷缩 | 沉闷 | 稍糊 | 稍凹 | 硬滑 | 否 |

# 最佳划分属性选择：信息增益

- 以属性“色泽”为例，其对应的 3 个数据子集分别为 $D^1$ (色泽=青绿)，$D^2$ (色泽=乌黑)， $D^3$ (色泽=浅白)
- 子集 $D^1$ 包含编号为 $\{1, 4, 6, 10, 13, 17\}$ 的 6 个样例，其中正例占 $p_1 = \frac{3}{6}$ ，反例占 $p_2 = \frac{3}{6}$ ，$D^2$, $D^3$ 同理，3 个结点的信息熵为：

$$\mathrm{Ent}(D^1) = -(\tfrac{3}{6}\log_2\tfrac{3}{6} + \tfrac{3}{6}\log_2\tfrac{3}{6}) = 1.000$$

$$\mathrm{Ent}(D^2) = -(\tfrac{4}{6}\log_2\tfrac{4}{6} + \tfrac{2}{6}\log_2\tfrac{2}{6}) = 0.918$$

$$\mathrm{Ent}(D^3) = -(\tfrac{1}{5}\log_2\tfrac{1}{5} + \tfrac{4}{5}\log_2\tfrac{4}{5}) = 0.722$$

- 属性“色泽”的信息增益为

$$\begin{aligned}\mathrm{Gain}(D, \text{色泽}) &= \mathrm{Ent}(D) - \sum_{v=1}^{3}\frac{|D^v|}{|D|}\mathrm{Ent}(D^v) \\ &= 0.998 - (\tfrac{6}{17}\times 1.000 + \tfrac{6}{17}\times 0.918 + \tfrac{5}{17}\times 0.722) \\ &= 0.109\end{aligned}$$

# 最佳划分属性选择：信息增益

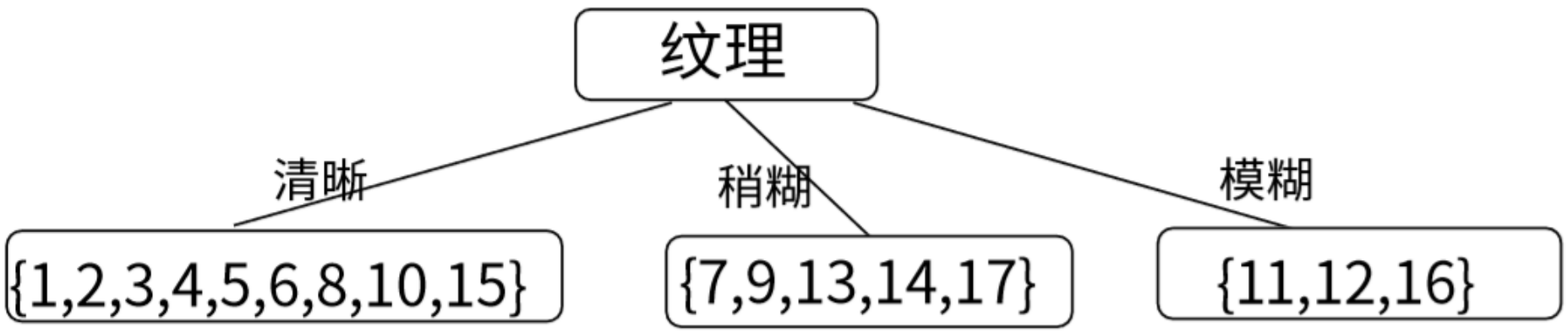
- 同样的方法，计算其他属性的信息增益为

$$\text{Gain}(D, \text{根蒂}) = 0.143 \qquad \text{Gain}(D, \text{敲声}) = 0.141$$

$$\text{Gain}(D, \text{纹理}) = 0.381 \qquad \text{Gain}(D, \text{脐部}) = 0.289$$

$$\text{Gain}(D, \text{触感}) = 0.006$$

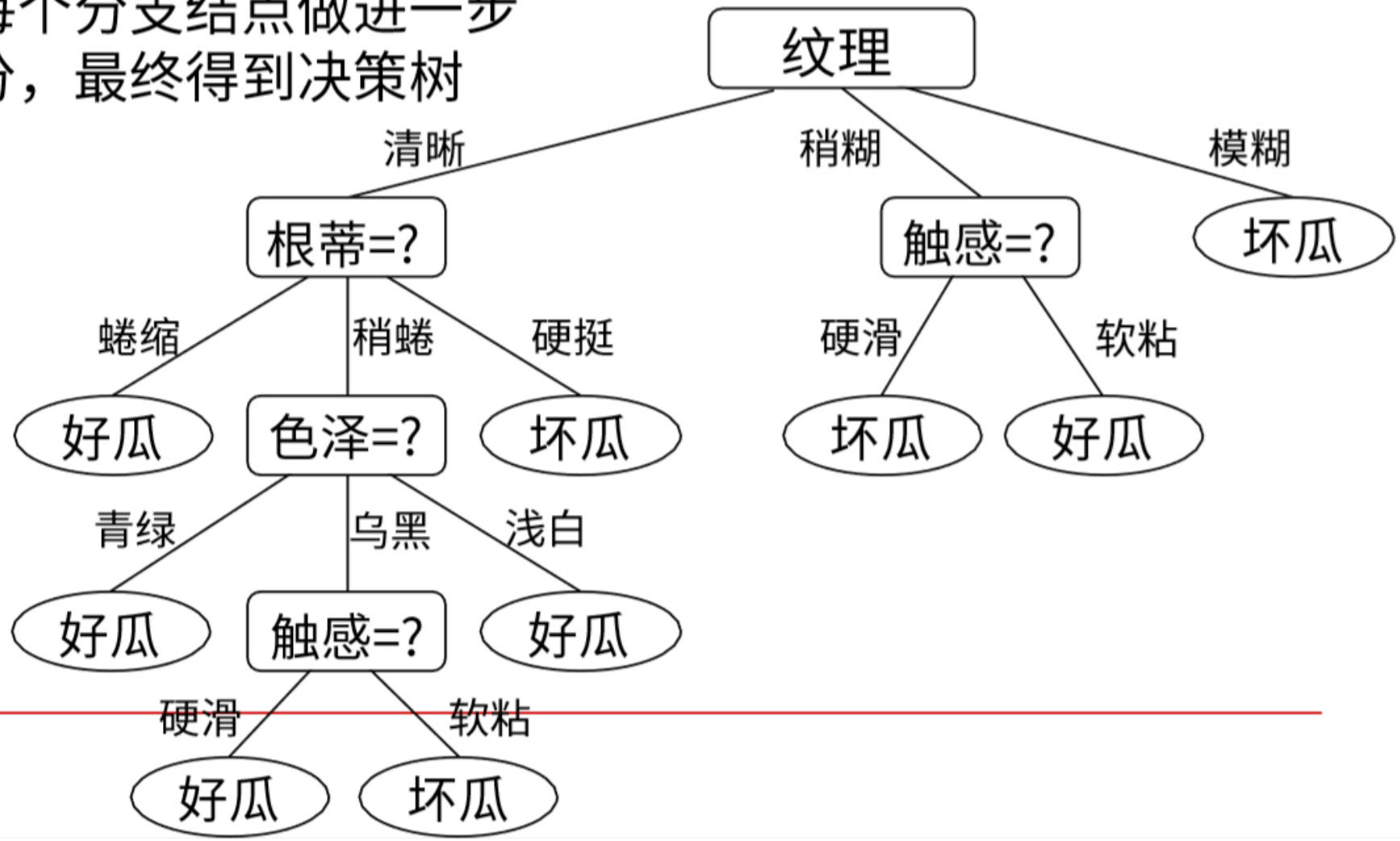
- 显然，属性“纹理”的信息增益最大，其被选为划分属性



# 最佳划分属性选择：信息增益

信息增益示例：

- 对每个分支结点做进一步划分，最终得到决策树



# 最佳划分属性选择：信息增益率

信息增益率 (gain ratio)：C4.5 中使用

信息增益的问题：对可取值数目较多的属性有所偏好

例如：考虑将“编号”作为一个属性

信息增益率：$$\text{Gain\_ratio}(D,a)=\frac{\text{Gain}(D,a)}{\text{IV}(a)}$$

其中 $$\text{IV}(a)=-\sum_{v=1}^{V}\frac{|D^v|}{|D|}\log_2\frac{|D^v|}{|D|}$$

属性a 的可能取值数目越多(即V 越大)，则IV(a) 的值通常就越大

启发式：先从候选划分属性中找出信息增益高于平均水平的，再从中选取增益率最高的

# 最佳划分属性选择：基尼指数

基尼指数 (gini index)：CART 中使用

$$\mathrm{Gini}(D) = \sum_{k=1}^{|\mathcal{Y}|} \sum_{k' \neq k} p_k p_{k'}$$

反映了从 D 中随机抽取两个样例，其类别标记不一致的概率

$$= 1 - \sum_{k=1}^{|\mathcal{Y}|} p_k^2 .$$

Gini(D) 越小，数据集 D 的纯度越高

属性a 的基尼指数： $\mathrm{Gini\_index}(D, a) = \sum_{v=1}^{V} \frac{|D^v|}{|D|} \mathrm{Gini}(D^v)$

在候选属性集合中，选取那个使划分后基尼指数最小的属性

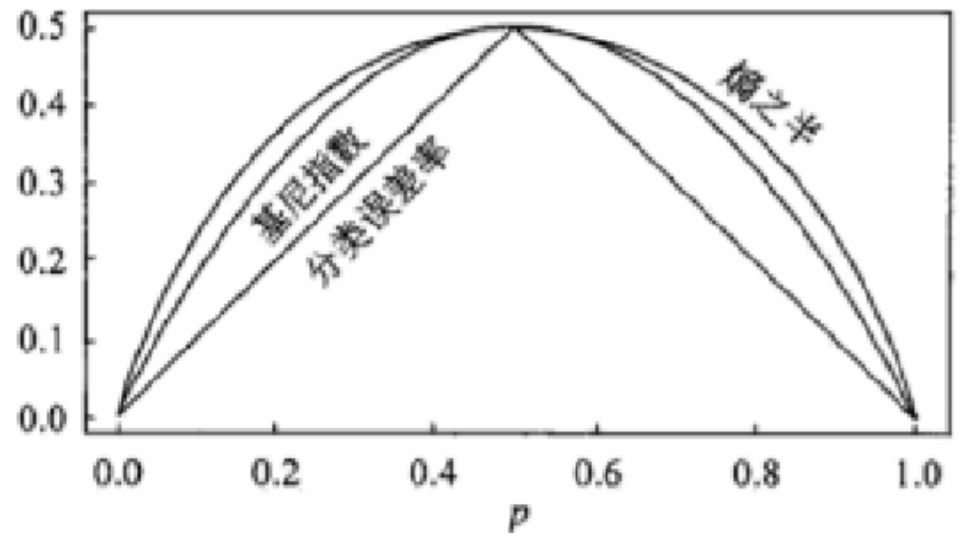
# 基尼指数 vs 熵 vs 分类错误率

基尼指数、熵、分类误差率三者之间的关系：

- 将f(x)=-lnx在x=1处一阶泰勒展开，忽略高阶无穷小，得到f(x)≈1-x

$$H(X)=-\sum_{k=1}^{K} p_k \ln p_k$$

$$\approx \sum_{k=1}^{K} p_k(1-p_k)$$



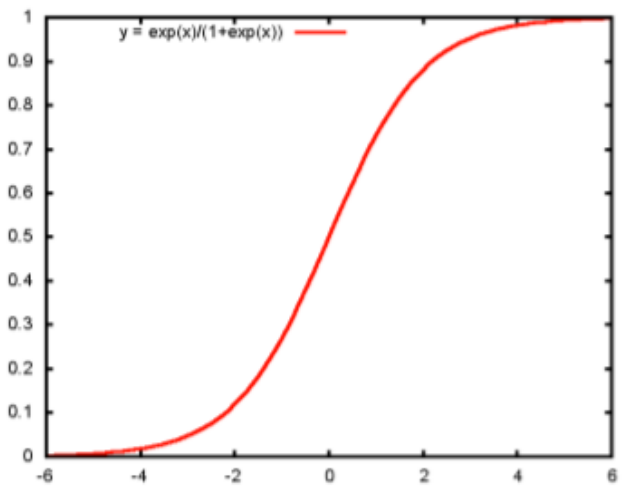
# 二分类

## 二分类视角看CART

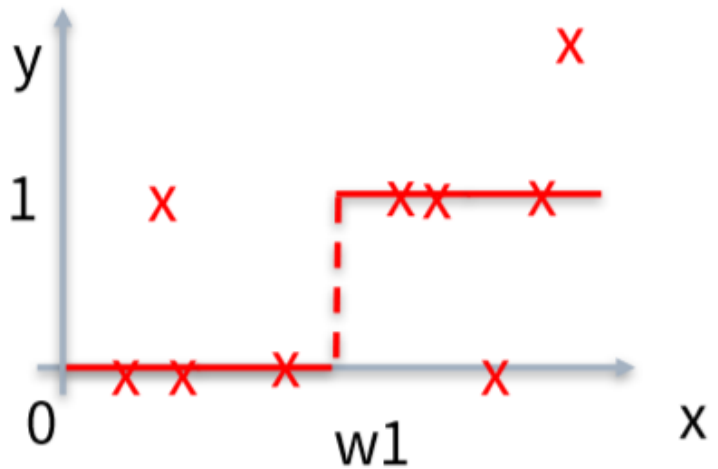
- 每一个产生分支的过程就是一个二分类过程
- 这个过程叫作“决策树桩”： decision stump
- 一棵CART是由许多决策树桩拼接起来的
- decision stump是只有一层的决策树

二叉树，每一个父结构只有两个直接的叶子节点。

逻辑回归




决策树桩



# 三种不同的决策树

- ID3：

  取值多的属性，更容易使数据更纯，其信息增益更大。

  训练得到的是一棵庞大且深度浅的树：不合理。

- C4.5

  采用信息增益率替代信息增益

- CART

  以基尼系数替代熵

  最小化不纯度，而不是最大化信息增益

# 联系与区别

决策树的分类规则都是互斥并且完备的。但CART是二叉树，而ID3、C4.5有几个值就划分为几个叶子节点。

CART既可以做分类，又可以做回归，而ID3、C4.5只是用于分类。

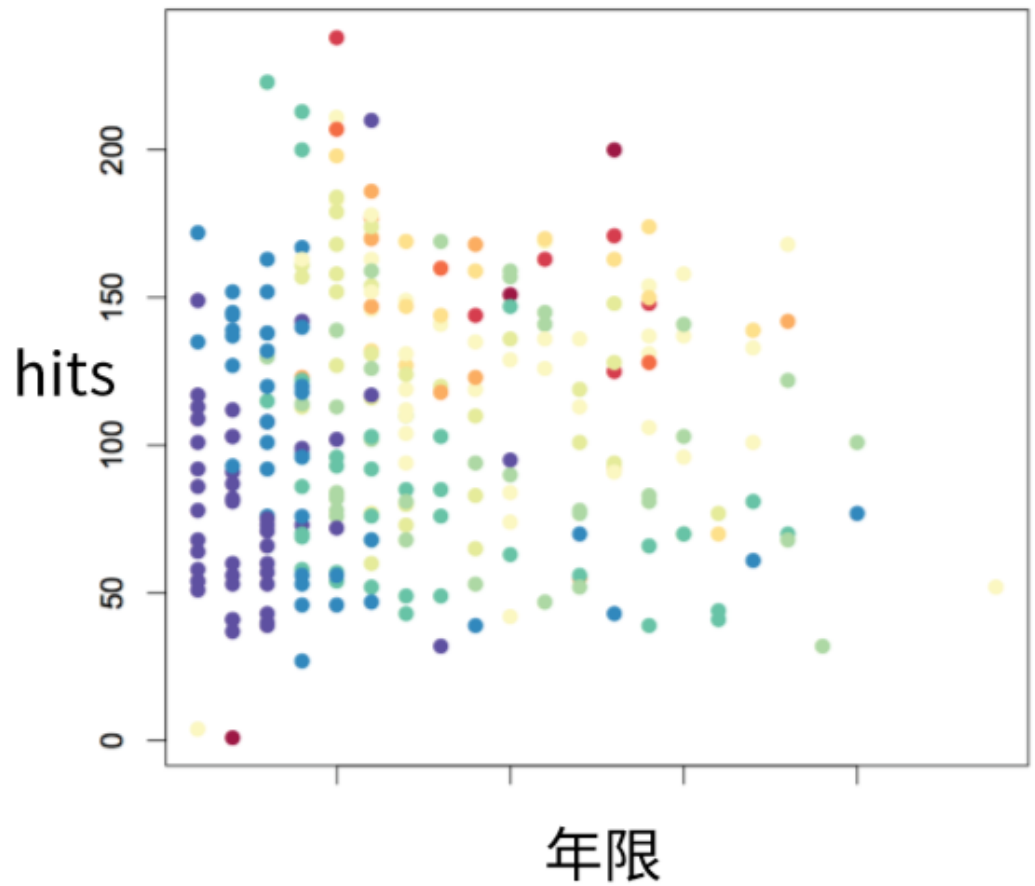
CART对于特征的利用是可以重复的，而作为分类的ID3、C4.5则是不能重复利用特征。

# 用决策树完成回归？

经典案例：

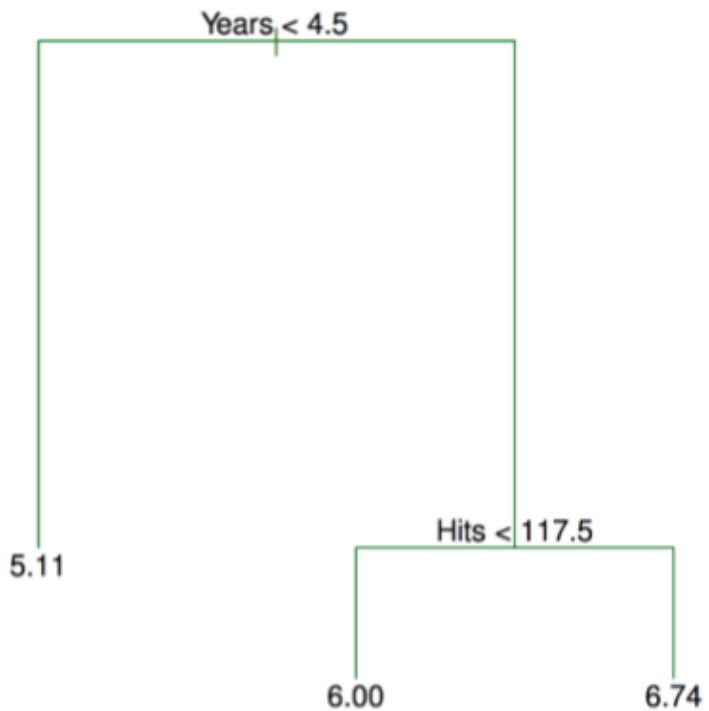
根据从业年限和表现，去预估棒球运动员的工资。

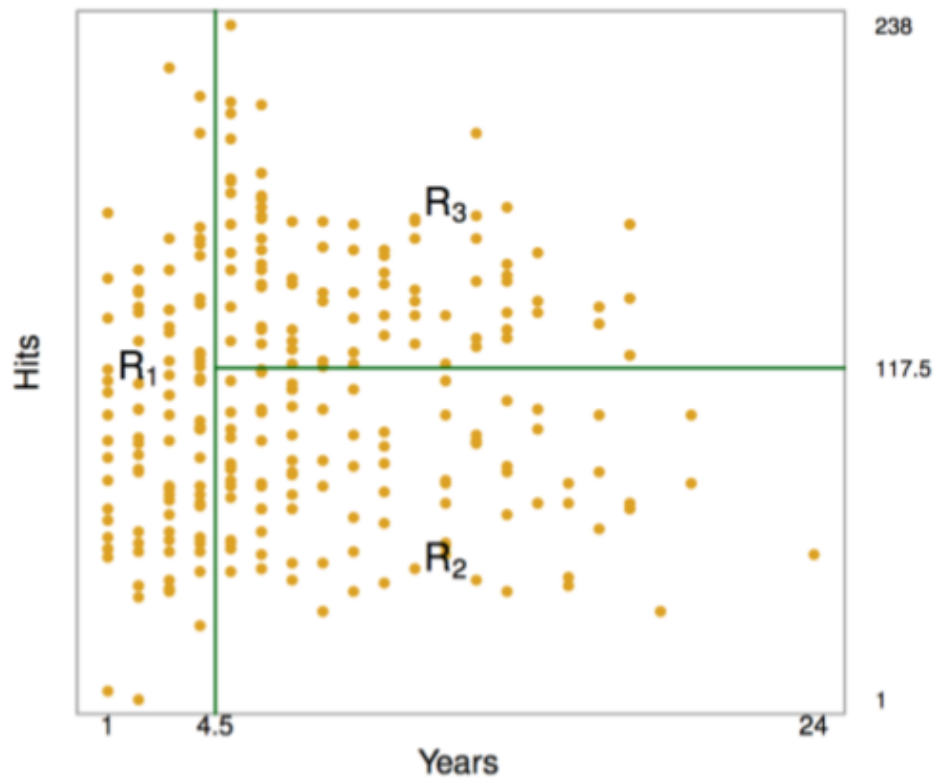
如右图所示，有1987个数据样本，包含322个棒球运动员。红黄表示高收入，蓝绿表示低收入。横坐标是年限，纵坐标是表现。



# 回归树

回归树背后的含义：
对空间的划分，整个平面被划分成3部分。
R1 = {X |Years < 4.5},
R2 = {X |Years ⩾ 4.5, Hits < 117.5}
R3 = {X |Years ⩾ 4.5, Hits ⩾ 117.5}



# 回归树

我们来正式介绍一下回归树的构建方法。

假设一个回归问题，预估结果$y \in R$，特征向量为$X = [x_1, x_2, x_3 \dots xp] \in R$，回归树的2个步骤是：

1. 把整个特征空间$X$切分成$J$个没有重叠的区域$R_1, R_2, R_3 \dots R_J$
2. 其中区域$R_J$中的每个样本我们都给一样的预测结果$\tilde{y}_{R_j} = \frac{1}{n}\sum_{j \in Rj} y_j$，其中n是$R_J$中的总样本数。

我们仔细观察一下上面的①过程，实际上我们希望能找到如下的RSS最小的化划分方式$R_1, R_2, R_3 \dots R_J$

$$RSS = \sum_{j=1}^{J} \sum_{i \in Rj} (yi - \tilde{y}_{R_j})^2$$

均方差最小化

但是这个最小化和探索的过程，计算量是非常非常大的。

我们采用探索式的递归二分来尝试解决这个问题。

# 回归树

递归二分

- 自顶向下的贪婪式递归方案

  - 自顶向下：从所有样本开始，不断从当前位置，把样本切分到2个分支里
  - 贪婪：每一次的划分，只考虑当前最优，而不回过头考虑之前的划分

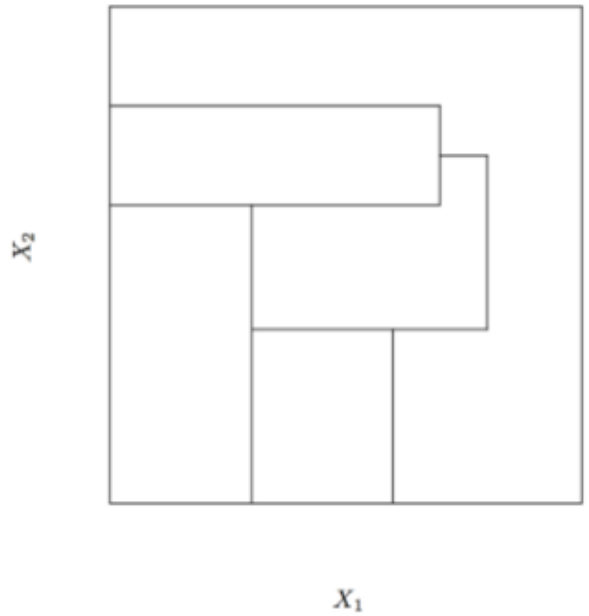
- 选择切分的维度(特征)$x_j$以及切分点$s$使得划分后的树$RSS$结果最小

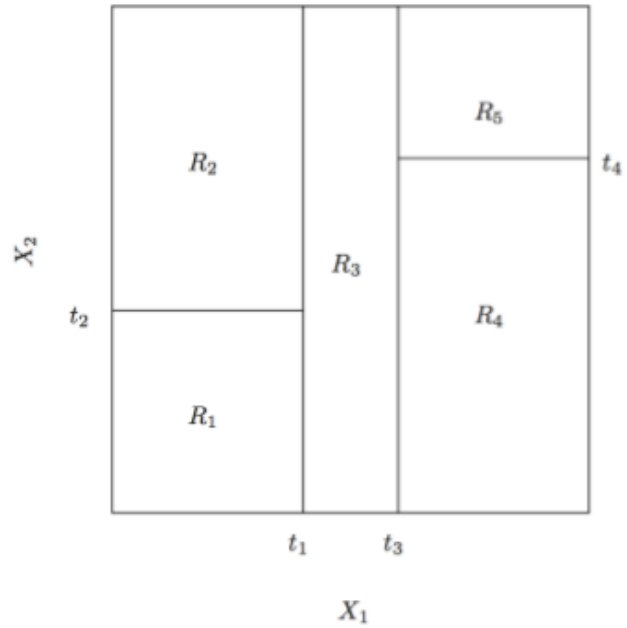
$$R_1(j,s) = \{x|x_j < s\}$$

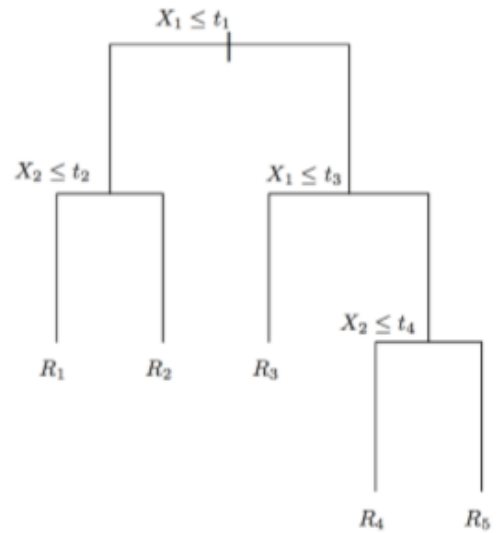
$$R_2(j,s) = \{x|x_j \geq s\}$$

$$RSS = \sum_{x_i \in R_1(j,s)} (y_i - \tilde{y}_{R1})^2 + \sum_{x_i \in R_2(j,s)} (y_i - \tilde{y}_{R_2})^2$$

# 回归树

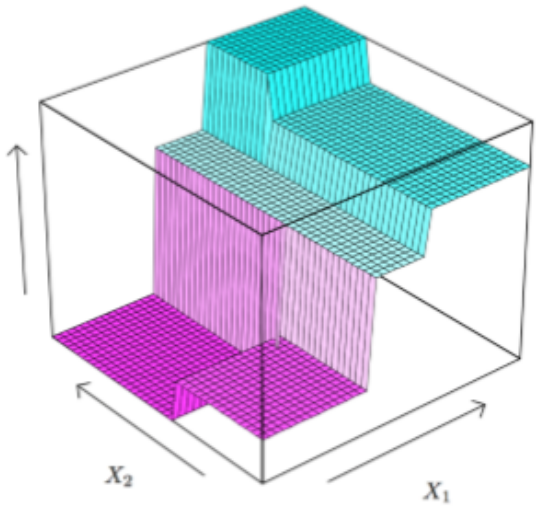



从左到右：

- 上左1：非二分切分得到的回归树空间划分
- 上左2：二分递归切分得到回归树空间划分
- 上左3：对应左2的回归树
- 下：对应左2的回归树空间划分与预估结果可视化



# 回归树

## 回归树剪枝

如果让回归树充分“生长”，同样会有过拟合的风险

- 解决办法：添加正则化项衡量
- 考虑剪枝后得到的子树$\{T_\alpha\}$，其中$\alpha$是正则化项的系数，当我固定一个$\alpha$后，最佳的$T_\alpha$就是使得下列式子值最小的子树。

$$\sum_{m=1}^{|T|}\sum_{x_i \in Rm}(y_i - \tilde{y}_{R_2})^2 + \alpha|T|$$

其中$|T|$是回归树叶子节点的个数

- 其中的$\alpha$可以通过交叉验证去选择

# Bagging思想

在介绍强大的RandomForest(随机森林)之前，我们先介绍一下Bootstraping和Bagging

Bootstraping: 名字来自成语“pull up by your own bootstraps”，意思是依靠你自己的资源，称为自助法，它是一种有放回的抽样方法，它是非参数统计中一种重要的估计统计量方差进而进行区间估计的统计方法。其核心思想和基本步骤如下：

(1) 采用重抽样技术从原始样本中抽取一定数量（自己给定）的样本，此过程允许重复抽样。

(2) 根据抽出的样本计算给定的统计量$T$。

(3) 重复上述N次（一般大于1000），得到$N$个统计量$T$。

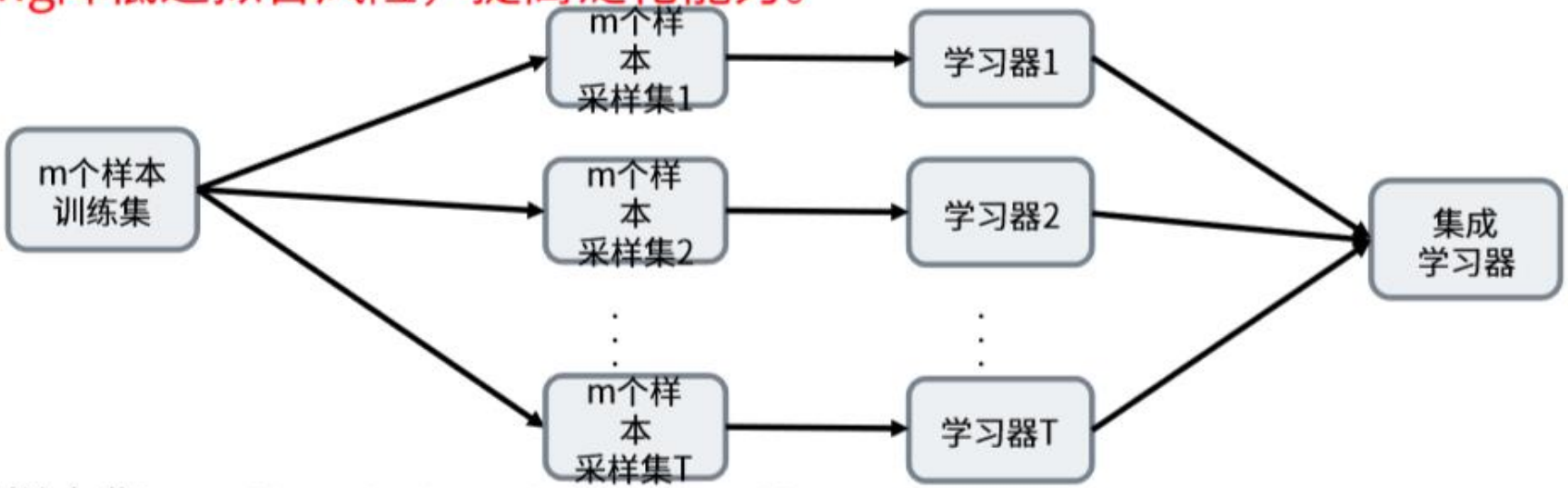
(4) 计算上述N个统计量T的样本方差，得到统计量的方差。

Bootstrap是现代统计学较为流行的一种统计方法，在小样本时效果很好。通过方差的估计可以构造置信区间等，其运用范围得到进一步延伸。

# Bagging思想

Bagging是bootstrap aggregating的缩写，使用了上述的bootstraping思想。

Bagging降低过拟合风险，提高泛化能力。




输入为样本集$D = \{(x, y_1), (x_2, y_2), \ldots (x_m, y_m)\}$

1）对于$t = 1,2 \ldots, T$:

a)对训练集进行第$t$次随机采样，共采集$m$次，得到包含$m$个样本的采样集$D_m$

b)用采样集$D_m$训练第m个基学习器$G_m(x)$

2) 分类场景，则T个学习器投出最多票数的类别为最终类别。回归场景，T个学习器得到的回归结果进行算术平均得到的值为最终的模型输出。

# Bagging思想

RandomForest(随机森林)是一种基于树模型的Bagging的优化版本。核心思想依旧是bagging，但是做了一些独特的改进。

RF使用了CART决策树作为基学习器，具体过程如下：

输入为样本集$D = \{(x, y_1), (x_2, y_2), \ldots (x_m, y_m)\}$

1）对于$t = 1,2 \ldots, T$:

a）对训练集进行第t次随机采样，共采集m次，得到包含m个样本的采样集$D_m$

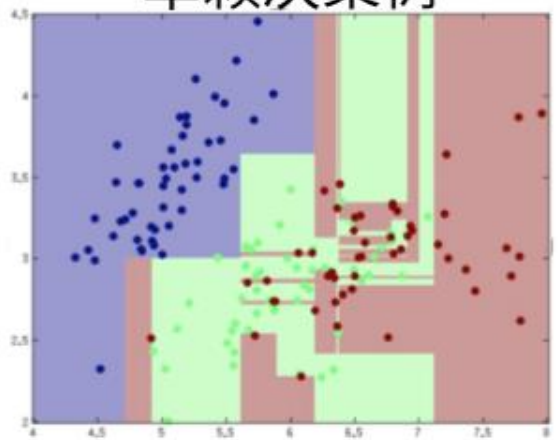
b）用采样集$D_m$训练第m个决策树模型$G_m(x)$，在训练决策树模型的节点的时候，在节点上所有的样本特征中选择一部分样本特征，在这些随机选择的部分样本特征中选择一个最优的特征来做决策树的左右子树划分

2) 分类场景，则T个基模型(决策树)投出最多票数的类别为最终类别。回归场景，T个基模型(回归树)得到的回归结果进行算术平均得到的值为最终的模型输出。

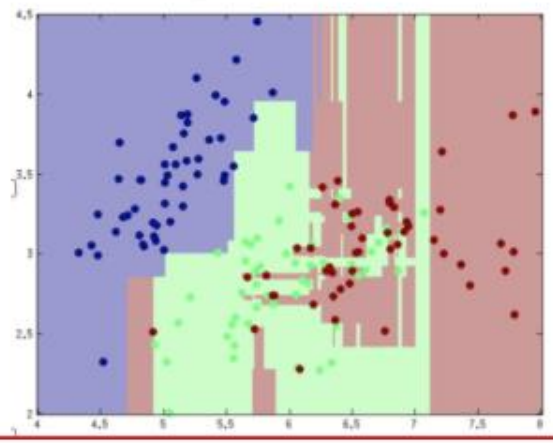
# Bagging思想

RandomForest（随机森林） 在iris数据集上的分类表现：

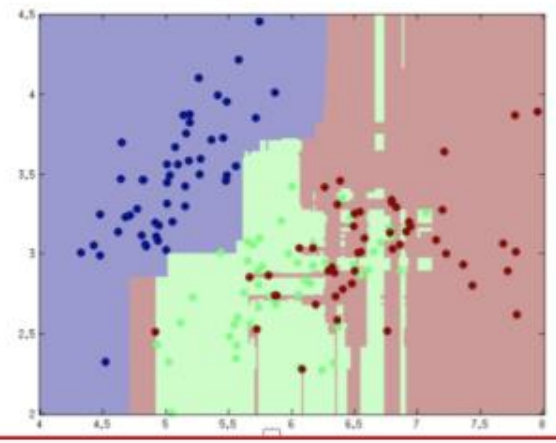
单颗决策树



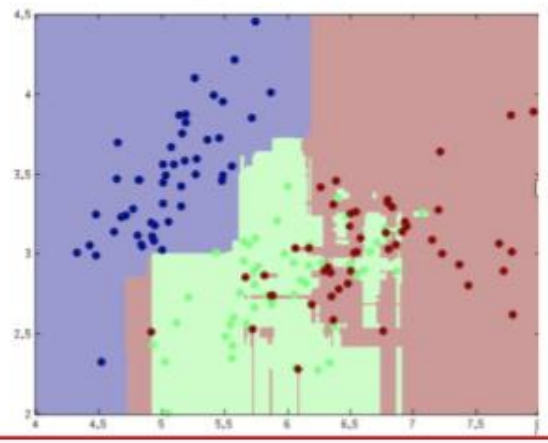
5颗决策树的随机森林



25颗决策树的随机森林



100颗决策树的随机森林



# 总结

1、熵

2、信息增益（ID3）

3、信息增益率（C4.5）

4、GINI系数（CART分类树）

5、均方误差（CART 回归树）

6、Bagging（随机森林）

感谢大家！

恳请大家批评指正！