树模型

1、决策树 ID3，C4.5，CART

2、随机森林RF

3、Adaboost

4、GBDT

5、XGboost

6、孤立森林（异常检测）

一、决策树

决策树是一种基本的分类和回归方法，用于分类主要借助每一个叶子节点对应一种属性判定，通过不断的判定导出最终的决策；用于回归则是用均值函数进行多次**二分**，用子树中数据的均值进行回归。决策树算法中，主要的步骤有：特征选择，建树，剪枝。下面对三种典型的决策树ID3，C4.5，CART进行三个步骤上的对比分析。

优点：

可解释性好，易可视化 ，特征工程中可用特征选择

样本复杂度$O(log(n))$，维度灾难

缺点：

易过拟合，学习最优模型N-P难，贪心搜索局部最优

虽然是非线性模型，但不支持异或逻辑

数据不均衡时不适合决策树

决策属性不可逆

一、特征选择

对于决策树而言，每一个非叶子节点都是在进行一次属性的分裂，选择最佳的属性，把不同属性值的样本划分到不同的子树中，不断循环直到叶子节点。其中，如何选择最佳的属性是建树的关键，决策树的一个特征选择的指导思想是**熵减**思想。常见的选择方式有ID3的信息增益，C4.5的信息增益率，CART的基尼指数，最小均方差。

这里分别介绍这ID3，C4.5，CART决策树的特征选择标准

1）信息增益

为了清楚的理解信息增益，先了解信息论中信息熵，以及条件熵的概念。熵是一种对随机变量不确定性的度量，不确定性越大，熵越大。

假设离散随机变量Y的概率分布为$P(Y)$，则其熵为：

$$H(Y) = -\sum_{y} P(y)logP(y)$$
$$= -\sum_{k=1}^{K} \frac{|C_k|}{|D|} log \frac{|C_k|}{|D|}$$

其中熵满足不等式$0 \leq H(Y) \leq log|Y|$。

在进行特征选择时尽可能的选择在属性X确定的条件下，使得分裂后的子集的不确定性越小越好（各个子集的信息熵和最小），即$P(Y|X)$的条件熵最小。

$$H(Y|X) = -\sum_{x,y} P(x,y)log(P(y|x))$$
$$= -\sum_{x_i \in X} \sum_{x_i,y} P(y,x_i)logP(y|x_i)$$
$$= -\sum_{i=1}^{n} \frac{|X_i|}{|X|} H(X_i)$$

其中X_i是表示属性X取值为i构成的子集。

信息增益表示的就是在特征X已知的条件下使得类别Y的不确定性减少的程度。即为特征X对训练数据集D的信息增益为$Gain(Y,X)$

$$Gain(Y,X) = H(Y) - H(Y|X)$$

用信息增益来进行特征选择，的确可以选择出那些对于类别敏感的特征。值得注意的是，信息增益没有考虑属性取值的个数的问题，信息增益倾向于选择取值较多的特征，因为划分的越细，不确定性会大概率越小。如对于数据表中主键这样的属性，用信息增益进行属性选择，那么必然会导致信息增益率最大，因为主键与样本是一一对应关系，而这样做分类是无意义的，即信息增益不考虑分裂后子集的数目问题。

2）信息增益比

针对信息增益偏向于选择特征值取值较多的特征，信息增益比将训练集关于特征X的信息熵作为分母来限制这种倾向（因为特征X的取值越多，表示特征X的信息熵会大概率大，不确定性越大）。即信息增益比定义为信息增益比上特征X的信息熵：

$$Gain_{ratio} = \frac{Gain(Y,X)}{H(X)}$$

其中$H(X) = -\sum_x P(x)logP(x)$。

3）基尼指数

基尼指数特征选择准则是CART分类树用于连续值特征选择，其在进行特征选择的同时会决定该特征的最优二分阈值。基尼指数是直接定义在概率上的不确定性度量：

$$Gini(D) = 1 - \sum_{k=1}^{K} p_k^2$$
$$= 1 - \sum_{i=1}^{K} \left(\frac{|C_k|}{|D|} \right)^2$$

可以看出，基尼指数与信息熵的定义极为一致，基尼指数去掉常数1负值部分直接采用概率，信息熵采用概率的对数。

4）最小均方差

最小均方差是针对回归树，回归问题一般采用最小均方差作为损失。在特征选择的同时会测试不同的二分阈值的最小均方误差，选择最有的特征和阈值。

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

其中c_i是第i个模型的回归值。

特征选择准则对比：

1）ID3采用信息增益，C4.5采用信息增益率，CART分类采用基尼指数，CART回归采用最小均方误差

2）信息增益，信息增益率，基尼指数都是用于分类树，最小均方误差用于回归树

3）信息增益，信息增益率同样可以处理连续值得情况，一般来讲连续值的处理只作二分，所以CART回归是一个标准的二叉树形式

二、建树

ID3和C4.5，CART分类树

输入：训练集D，特征集A,阈值ϵ

输出：决策树T

1、若D中所有样本属于同一类C_k，T为叶子节点，并将该叶子节点的类别标记为C_k，返回上一次递归。

2、若$A=\varnothing$，即所有的属性都使用完了，T为叶子节点，并把该子集中最多一类C_k标记为该叶子节点的类别，返回上一次递归。否则，3）

3、进行特征选择。选择信息增益(信息增益比,基尼指数)最大的特征A_g，如果A_g的信息增益（信息增益率,基尼指数）小于预设的阈值ϵ，同样T为叶子节点，并把该子集中最多一类C_k标记为该叶子节点的类别，返回上一次递归。否则，4）

4、符合建树要求。对A_g的每一个可能值a_i，依次$A_g=a_i$将D分割为若干个非空子集D_i，将D_i中实例最多的类别标记为该节点的类别C_k，依次以D_i为样本集，$A-A_g$为特征集，递归的调用（1-4）步，直到结束。

CART回归树

由于回归树不存在剩余样本属于一类或者特征用完的情况，所以分类树没有前面的前面两步。

1 对每一个特征进行步长（样本）循环搜索不同阈值下的最小的均方误差记为该特征的均方误差，从所有特征的均方误差选择出最小均方误差。如果最小均方误差小于预设的最小误差，或者分裂后的子集的样本数小于预设的最小值则进行建立叶子节点T，返回上一次递归。

2否则，以特征A_g作为分裂属性，根据阈值ϕ进行二分，建立左右子树，建立线性回归模型。递归（1-2）步，直到结束。

分类树和回归树建树区别：

1）回归树中特征可以重复进行选择，而分类树的特征选择只能用一次

2）回归树比分类树少了特征集合为空，样本集合同属一类这两个返回标志，只能人工干预（指标无提升）

3）对于特征下划分阈值的分裂，一般只作二分裂，不然就成了密度估计问题了

三、剪枝

决策树生成算法递归生成的决策树，按照建树的过程直到结束。这样产生的决策树往往对训练集的分类很准确，但是对未知数据却没有那么准确，容易出现过拟合现象。因此，决策树需要借助验证集来进行剪枝处理，防止决策树过拟合。

预剪枝

预剪枝是在构造决策树的过程中，对比属性划分前后决策树在验证集上是否由精度上的提高。由于非叶子节点中的样本往往不属于同一类，采用多数样本标记为该节点的类别进行决策。若划分后的决策树在精度上有提高，则正常分裂，否则进行剪枝。其过程是一个贪心过程，即每一次划分都必须使得决策精度有所提高。当然也可以对建树进行约束，比如信息增益小于一定阈值的情况或者建树之后其中一个子集的样本数小于一定数量进行预剪枝，统计学习方法书中采用熵加叶子节点树作为损失函数来控制预剪枝。

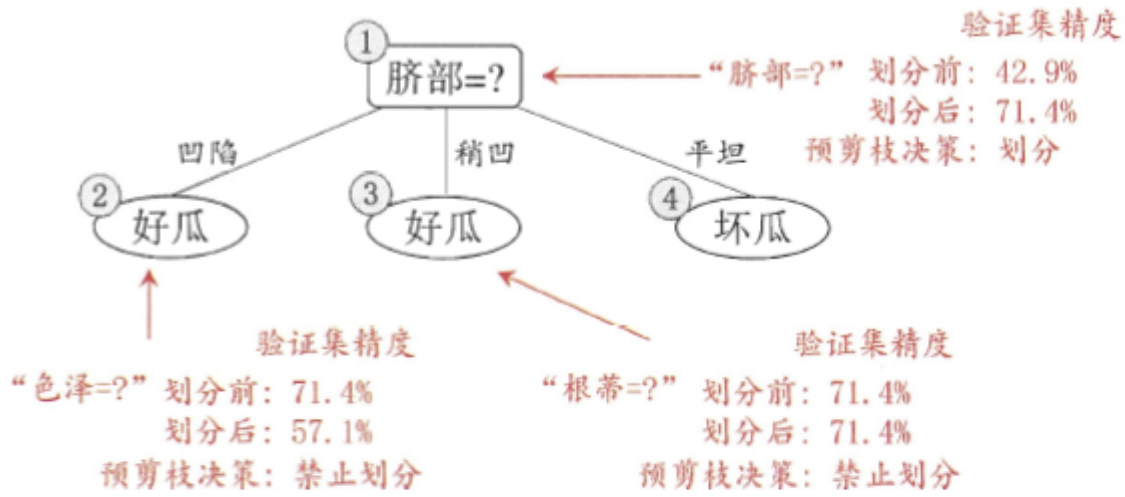


图 4.6 基于表 4.2 生成的预剪枝决策树

后剪枝

后剪枝是在决策树建立后以后，自底向上的对决策树在验证集上对每一个非叶子节点判断剪枝前和剪枝后的验证精度，若剪枝后对验证精度有所提高，则进行剪枝。（不同预剪枝的是，预剪枝是对划分前后的精度进行比较，而后剪枝是对剪枝前和剪枝后的验证精度进行比较），相对于预剪枝，后剪枝决策树的欠拟合风险小，泛化能力优于预剪枝，但时间开销大。

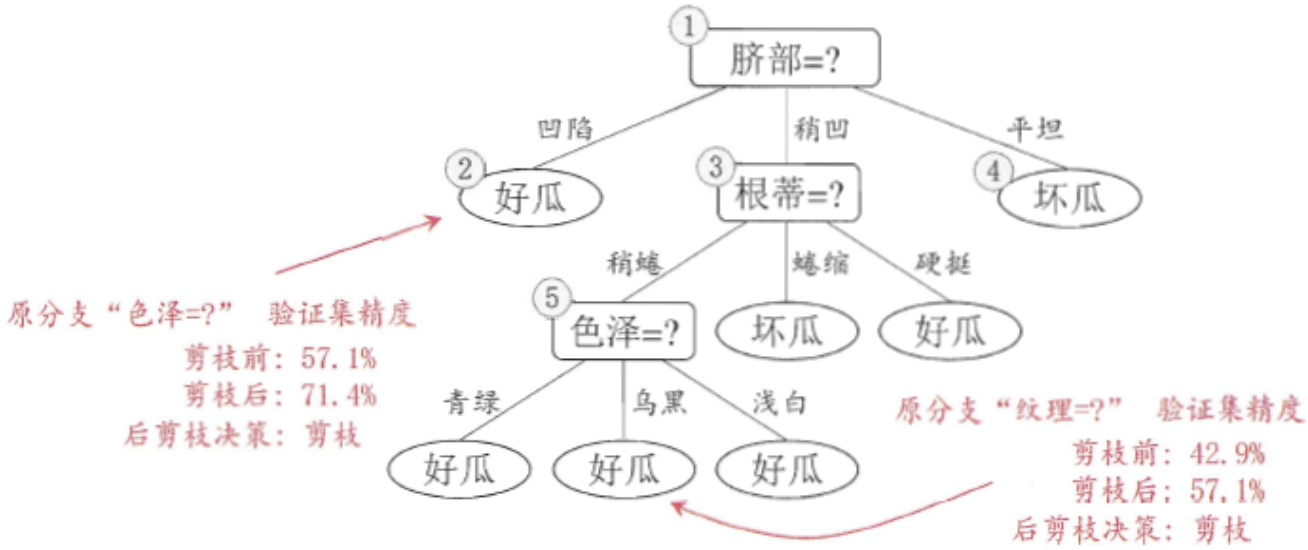


图 4.7 基于表 4.2 生成的后剪枝决策树

决策树总结：

1）ID3,C4.5,CART没有必要严格按照特征选择方式，建树过程严格划分

2）决策树选择从问题出发，如果是回归问题，可以采用CART回归树，如果是分类问题那么采用分类树

3）决策树选择从数据出发，如果属性是连续值，二分离散化建二叉树，如果属性是离散值，则建多叉树

4）特征选择准则可以互用，一般来讲连续值得特征可以反复选择，而离散值的特征只能用一次

二、随机森林RF

一、集成学习

集成学习通过构建多个学习器采用加权的方式来完成学习任务，类似于“三个臭皮匠顶个诸葛亮”的思想。当然多个学习器之间需要满足一定的条件，一般来讲，多个学习器同属于一种模型，比如决策树，线性模型，而不会交叉用多种模型。为了保证集成学习的有效性，多个弱分类器之间应该满足两个条件：

1）准确性：即个体学习器要有一定的准确性，在训练集上正确率至少达到 0.5 才能有好的效果。

2）多样性：即学习器之间要有一些差异，因为完全相同的几个学习器集成起来后完全没有任何效果。

目前，集成学习主要分为Bagging和Boosting两种方式，前者通过Booststrap Aggregation的重采样得到多组训练集，并行的训练基学习器。而后者是一种提升的思想，基学习器是串行执行的，下一个学习器会基于上一个学习的经验进行调整，学习器前后有依赖关系，多个学习器最终组合得到强学习器。

//集成学习的有效性说明：

二、随机森林

随机森林是集成学习中Bagging方式的代表，其相对决策树的提高很重要的一点防止过拟合，主要通过以下两点来防止过拟合，这与深度学习中的Dropout（随机的丢失一些样本和特征）技术非常相似

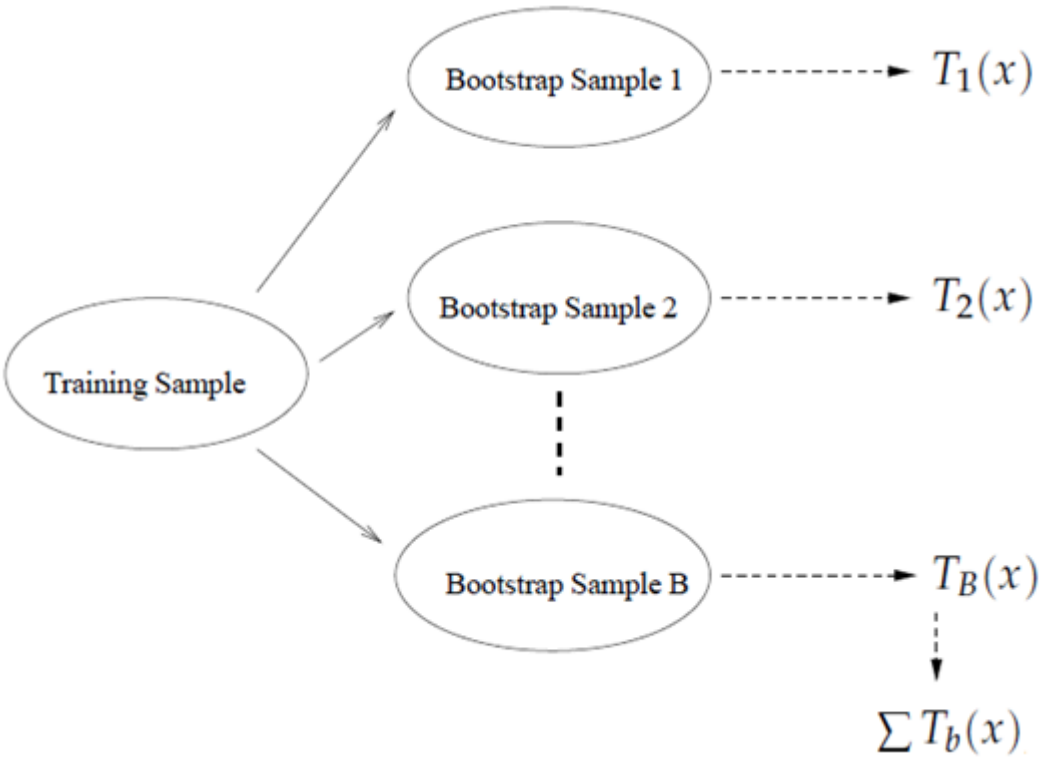
1）样本选择随机：Bootstrap Sampling

2）特征选择随机：基学习器决策树的特征选择$log_2 d$

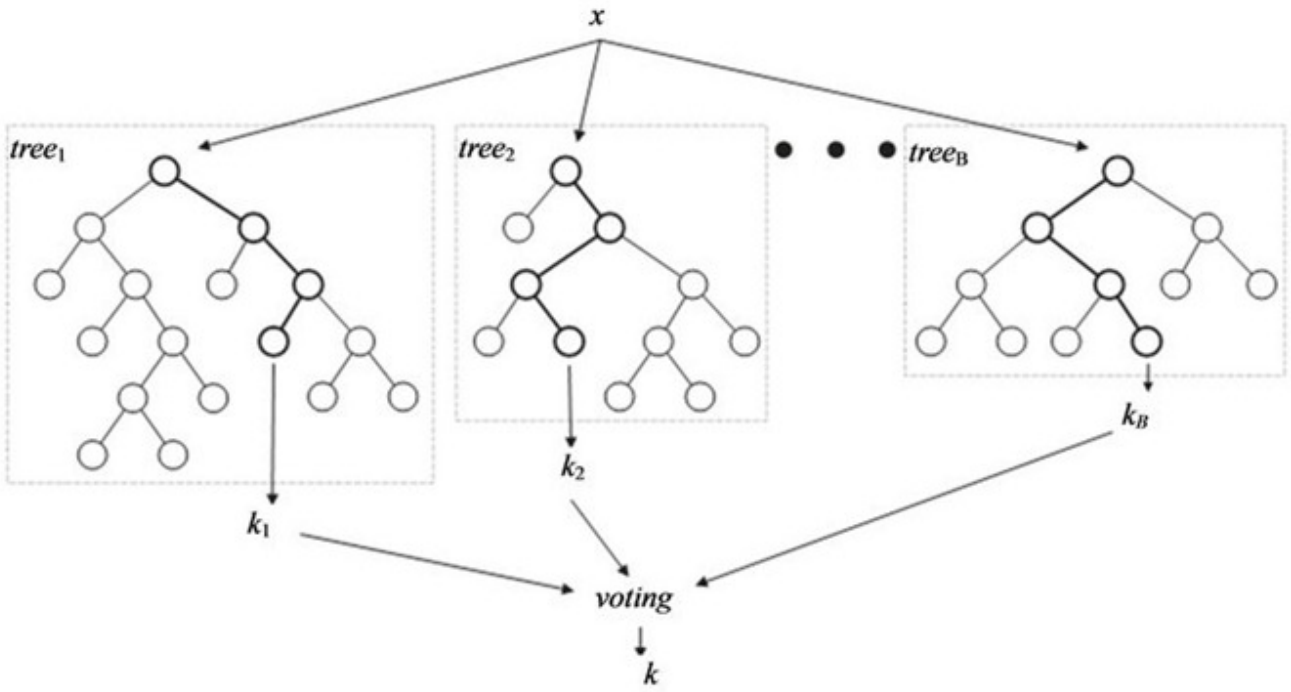
Bootstrap Sampling：是一种统计学上的抽样方法，该方法是这样执行的，对于有m个样本的数据集D，进行m次有放回采样得到数据集 D'，这样D与D'的大小一致。有放回采样使得D'中有的样本重复出现，有的样本则没有出现，简单估计一下，某个样本在m次采样中始终没被采到的概率为$(1-\frac{1}{m})^m$，取极限：

$$\lim_{m\to\infty}(1-\frac{1}{m})^m=\frac{1}{e}\approx 0.368$$

即D中的样本大概有63.2%几率出现在D'中，采样出B个Bootstrap 样本集 $D_1, D_2, \ldots, D_B$，对这K个样本集分别训练一个基学习器 $T_b(x)$，结合这些基学习器共同作出决策。决策时，在**分类任务中通常采用投票法**，若两个类别票数一样，最简单的做法是随机选择一个；而**回归任务则一般使用平均法**。整个流程如下所示：



基学习器：早期的Bagging方法是每个基学习器都是一个决策树，完全按照决策树的规则建树。随机森林则在Bagging的基础继续采用特征随机，每个基学习器只对在k个特征构成的子集下进行建树，一般取$k=log_2 d$。这样构建的决策树相对于完整的决策树是一个“浅决策树”，这样就构成了特征的随机性。



随机森林过程：

1. 假设我们设定训练集中的样本个数为N，然后通过Bootstrap Sampling来获得B个有重复的样本集N'；
2. 针对每个样本集N'独立训练，对于有d个特征的数据集，随机选择$k(k<d)$个特征构成特征选择集Ψ。然后在样本集N'，特征集Ψ上构建决策树。k值是保持不变的， 随机选取特征增加树的独立性，每棵决策树都最大可能地进行生长而不进行剪枝；
3. 通过对所有的决策树进行加权来预测新的数据（在分类时采用多数投票，在回归时采用平均）。

到此，随机森林基本介绍完，但是依然存在问题，随机森林为什么能防止过拟合，随机森林适合什么样的场景？

Bias and Variance 分析

从Bias和Variance的角度分析，Bagging对样本的重采样得到B个N'训练集，对于每个训练集训练一个基学习器，因为基学习器相同，因此各个学习器有近似的Bais和Variance（学习器并不一定独立）。假设每个学习器的权重相同即$\frac{1}{B}$。每个学习器的损失用L_b表示，那么随机森林的损失可表示为：

$$E(\frac{1}{N}\sum_b L_b) = E(L_b)\ ,\ b=1,2,\dots,B$$

所以 Bagging 后的 Bias和单个基学习器的接近，并不能显著降低bias，但是若各基学习器独立，因为每个学习器的权重是$\frac{1}{B}$，所以引入的方差为$\frac{1}{B}Var(X)$，那么随机森林的Variance可表示为：

$$Var(\bar{X}) = B\left(\frac{1}{B}Var(X)\right)^2\ ,\ b=1,2,\dots,B$$

可以看出，Bagging通过降低Variance来防止过拟合，严格来说每个学习器之间不严格独立，所以Variance的降低会小于B倍。

随机森林的优点：

> 1. 正如上文所述，随机森林在解决分类与回归两种类型的问题有很大的优势
> 2. 随机森林抗过拟合能力比较强

3. 随机森林能处理很高维度的数据（也就是很多特征的数据），并且不用做特征选择，因为建树时会随机选择一些特征作为待选特征子集
4. 训练速度快，容易做成并行化方法(训练时，树与树之间是相互独立的)
5. 随机森林可以做类似于GBDT那样的特征组合；
6. 在对缺失数据进行估计时，由于随机丢失特征，随机森林依然十分有效；
7. 当存在分类不平衡的情况时，随机森林能够提供平衡数据集误差的有效方法，比如对于 10:1 的数据，将多数数据分为 10份，做 10个 1:1 的单模型然后 Bagging 起来即可。

随机森林的缺点：

1. 随机森林在解决回归问题时，并没有像它在分类中表现的那么好。因为它并不能给出一个连续的输出。当进行回归时，随机森林不能够做出超越训练集数据范围的预测，这可能导致在某些特定噪声的数据进行建模时出现过度拟合。（PS:随机森林已经被证明在某些噪音较大的分类或者回归问题上会过拟合）。
2. 对于许多统计建模者来说，随机森林给人的感觉就像一个黑盒子，你无法控制模型内部的运行。只能在不同的参数和随机种子之间进行尝试。
3. 可能有很多相似的决策树，掩盖了真实的结果。
4. 对于小数据或者低维数据（特征较少的数据），可能不能产生很好的分类。（处理高维数据，处理特征遗失数据，处理不平衡数据是随机森林的长处）。

三、Adaboost

一、Adaboost的Boosting理解

Adaboost是集成学习中Boosting方式的代表。多个基学习器其串行执行，下一个学习器基于上一个学习器的经验，通过调整样本的权重，使得上一个错分的样本在下一个分类器更受重视而达到不断提升的效果。Adaboost集成多个学习器的关键在两点：

1）设置基学习器的权重 $a_m = \frac{1}{2} log \frac{1-e_m}{e_m}$

2）调整样本的权重$w_{m+1,i} = \frac{w_{m,i}}{Z_m} exp(-a_m y_i G_m(x_i))$

相对随机森林的**Bootstrap Sampling**重采样技术，可以看出Adaboost的权重调整是有目的性，是基于上一个学习器的经验，这也导致Adaboost在基学习器层是串行的。另外值得探讨的是权重为何如此设置？

Adaboost算法采用的基学习器是**二值函数（二叉树）**模型（当然Adaboost的核心是采用Boosting的思想），下面先来看Adaboost算法的整个流程，后面分析Adaboost在设计上巧妙之处

输入： 训练集 $D = \{(x_1 y_1), (x_2, y_2), \dots, (x_N, y_N)\}$，其中$x_i \in \mathbb{R}^n$，$y_i \in \{+1, -1\}$

输出：强学习器$G(x)$

1）初始化权值 ： $D_1 = w_{11}, w_{12}, \dots, w_{1N}$，$w_{1i} = \frac{1}{N}, i = 1, 2, \dots, N$

2）训练M个基学习器，对$m = 1, 2 ... M$

a）使用权值分布D_m与相应的基学习器算法得到第m个基学习器$G_m(x): x_i \rightarrow y_i$

b）计算基学习器$G_m(x)$的训练误差：

$e_m = P(G_m(x_i) \neq y_i) = \sum_i^N w_{mi} I(G_m(x_i) \neq y_i) / \sum_i^N w_{mi}$

c) 计算基学习器的权重

如果$e_m > \frac{1}{2}$：$a_m = 0$，舍弃基学习器，样本权重更新与不更新一致

否则：$a_m = \frac{1}{2}\log\frac{1-e_m}{e_m}$

d) 更新样本的权重：

$D_{m+1} = (w_{m+1,1}, \ldots, w_{m+1,i}, \ldots, w_{m+1,N})$

其中，$w_{m+1,i} = \frac{w_{mi}}{Z_m}\exp(-a_m y_i G_m(x_i))$

这里Z_m是归一化因子：$Z_m = \sum_i w_{mi}\exp(-a_m y_i G_m(x_i))$，使得$D_{m+1}$满足一个概率分布

3) 得到M个基学习器之后，将基学习器线性组合：

$f(x) = \sum_m a_m G_m(x)$

4) 得到最终的分类器：

$G(x) = sign(f(x)) = sign(\sum_m a_m G_m(x))$

Adaboost算法流程基本与Boosting思想一致，特别之处在于权重的设计，下面分析一下：

1) 基学习器的权重$a_m = \frac{1}{2}\log\frac{1-e_m}{e_m}$，当$0 \le e_m \le \frac{1}{2}$，$a_m \ge 0$，且$a_m$随$e_m$的减小而增大，也就是说当基学习器的误差越小，权重越大。

2) 样本权重更新公式可以表示如下：

$$w_{m+1,i} = \begin{cases} \frac{1}{Z_m} w_{mi} e^{-a_m}, G_m(x_i) = y_i \\ \frac{1}{Z_m} w_{mi} e^{a_m}, G_m(x_i) \ne y_i \end{cases}$$

也就是说正确分类$y_i = G_m(x_i)$，那么$-a_m < 0$，正确分类的样本权重在上一次的基础上乘上一个小于1的因子而减小，反之，错分的样本的权重增大。

3) M个基学习器在线性组合时，需要注意的是$\sum_m a_m \ne 1$，最终的$f(x) = \sum_m a_m G_m(x)$是一个$[-\sum_m a_m, \sum_m a_m]$区间的值，符号决定分类，绝对值表示分类一个确信度。

二、Adaboost的指数损失理解

Adaboost算法是前向分步加法算法的特例，以模型为加法模型，损失函数为指数函数的二类分类学习方法。考虑加法模型（additive model）

$$f(x) = \sum_m \beta_m b(x;\gamma_m)$$

其中，$b(x;\gamma_m)$为基函数，γ_m为基函数的参数，β_m为基函数的权重，显然这是一个加法模型。

在给定训练集和损失函数$L(y, f(x))$的条件下，学习加法模型$f(x)$就是最小化损失函数的问题：

$$arg\min_{\beta_m,\gamma_m} \sum_{i=1}^{N} L\left[y_i, \sum_m \beta_m b(x_i;\gamma_m)\right]$$

当然，我们可以将加法模型看作一个复合函数（加法），直接优化各个系数和基函数参数，但这样问题就变复杂了。考虑前向分步算法，逐个优化每一个基函数和系数来逼近复合函数，那么问题就简化了。具体的，每一步需要优化如下目标函数：

$$arg \min_{\beta_m,\gamma_m} \sum_{i=1}^{N} L\left[y_i, \beta_m b(x_i;\gamma_m)\right]$$

按照这种分步策略，每步优化一个基函数和系数，我们有前向分步算法如下：

输入：训练集$D=\{(x_1 y_1),(x_2,y_2),\ldots,(x_N,y_N)\}$，损失函数：$L(y,f(x))$，基函数$\{b(x;\gamma)\}$

输出：加法模型$f(x)$

1)初始化$f_0(x)=0$

2)学习M基函数和系数，从$m=1,2,\ldots M$

a)极小化损失函数 $\beta_m,\gamma_m = arg\min_{\beta,\gamma}\sum_i L(y_i, f_{m-1}(x_i)+\beta b(x_i;\gamma))$

b)更新$f_m(x)=f_{m-1}(x)+\beta_m b(a_i;\gamma_m)$

3)得到最终的加法模型$f(x)=f_M(x)=\sum_m \beta_m b(x,\gamma_m)$

前向分步算法通过逐个优化基函数，逐渐弥补残差的思想完成M个基函数的学习。

回到**Adaboost**算法，Adaboost是前向分步加法模型的特例。特例在于Adaboost是二分类，且损失函数定义为指数损失和基函数定义为二分类函数。

1）指数损失函数$L(y,f(x))=-yexp(f(x))=exp(-yf(x))$，$y\in\{-1,1\}$

2）基函数 $G_m(x):x_i\rightarrow y_i, y_i\in\{-1,1\}$

在Adaboost算法中，我们最终的强学习器为：

$$f(x)=\sum_m a_m G_m(x)$$

以第m步前向分步算法为例，第m个基函数为：

$$f_m(x)=f_{m-1}(x)+a_m G_m(x)$$

其中$f_{m-1}(x)$为：

$$f_{m-1}(x)=f_{m-2}(x)+a_{m-1}(x)G_{m-1}(x)=a_1 G_1(x)+,\ldots,+a_m G_m(x)$$

根据前向分步算法得到a_m和$G_m(x)$使得$f_m(x)$在训练集D上的指数损失最小，即：

$$\begin{aligned} a_m^*, G_m^*(x) &= arg\min_{a,G}\sum_i exp[-y_i(f_{m-1}(x_i)+a_m G_m(x_i))] \\ &= arg\min_{a_m,G_m}\sum_i exp[-y_i f_{m-1}(x_i)]\cdot exp[-y_i a_m G_m(x_i)] \\ &= arg\min_{a_m,G_m}\sum_i w_{mi} exp[-y_i a_m G_m(x_i)] \end{aligned}$$

其中$w_{m,i}=exp[-y_i f_{m-1}(x_i)]$，可以看出$w_{m,i}$只与$f_{m-1}(x)$有关，与当前的学习器无关。由$f_m(x)=f_{m-1}(x)+a_m G_m(x)$，我们可以得出$w_{m,i}$的另外一种表示：

$$w_{m,i} = w_{(m-1,i)} exp[-y_i a_m G_m(x)]$$

这也就是样本权重更新的表达式（未归一化）。

现在分析目标函数，首先看$G_m^*(x)$，因为$a_m > 0$,$G_m(x) = \{-1, 1\}$，要使目标函数取到最小值，那么必然有：

$$G_m^*(x) = arg \min_{G_m} \sum_i w_{mi} I(y_i \neq G_m(x_i))$$

也就是说$G_m^*(x)$是第m步使得样本加权训练误差最小的基分类器。将$G_m^*(x)$带入目标函数有：

$$\begin{aligned}
&\sum_i w_{mi} exp[-y_i a_m G_m(x_i)] \\
&= \sum_{y_i = G_m(x_i)} w_{mi} e^{-a_m} + \sum_{y_i \neq G_m(x_i)} w_{mi} e^{a_m} \\
&= \sum_{y_i = G_m(x_i)} w_{mi} e^{-a_m} + \sum_{y_i \neq G_m(x_i)} w_{mi} e^{a_m} + \sum_{y_i \neq G_m(x_i)} w_{mi} e^{-a_m} - \sum_{y_i \neq G_m(x_i)} w_{mi} e^{-a_m} \\
&= e^{-a_m} \sum_i w_{mi} + (e^{a_m} - e^{-a_m}) \sum_{y_i \neq G_m(x_i)} w_{mi}
\end{aligned}$$

上式对a_m求导即可，使得倒数为0，即可得到 a_m^* .

$$a_m^* = \frac{1}{2} log \frac{1 - e_m}{e_m}$$

其中e_m是第m个基分类器$G_m(x)$的分类错误率：

$$e_m = \frac{\sum_{y_i \neq G_m(x_i)} w_{mi}}{\sum_i w_{mi}} = \sum_i w_{mi} I(y_i \neq G_m(x_i))$$

最后来看Adaboost的权重调整，都是前向分步算法基于一个目标：

$$\min_{a,G} \sum_i exp[-y_i (f_{m-1}(x_i) + a_m G_m(x_i))]$$

可见一切设计都不是偶然，都是必然。Adaboost只是前向分步加法模型的特例，是GBDT的二分类特例，采用牛顿法求解版。

四、GBDT

提升树，GBDT同样基于最小化第m个学习器和前$m-1$个学习器累加起来损失函数最小，提升树采用残差的思想来最小化损失函数，将投票权重放到学习器上，使得基学习器的权重都为1；GBDT将损失用一阶多项式拟合，基学习器拟合梯度，学习器的权重为一阶多项式的系数。

在前面的Adaboost中，我们需要学习M个基学习器，赋予不同的权重组合得到最后的强学习器。它是基于M个基学习器组合而成。而提升树中，直接将他们以“残差（损失函数的残差）”的形式累加起来，故也为加法模型，而且是逐步累加。

提升树模型如下：

$$f_M(x) = \sum_{m=1}^{M} T(x; \theta_m)$$

其中，$T(x,\theta_m)$表示决策树，θ_m为决策树的参数，M为树的个数。

提升树优化过程：

输入：训练集$\{(x_i,y_i)\}_{i=1}^N$，损失函数$L(y,f(x))$

输出：提升树$f_M(x)$

1)初始化$f_0(x)=0$

2)对$m=1,2,...M$

a) 计算残差:

$$r_{mi}=y_i-f_{m-1}(x_i)\quad ,\{1,2,...,N\}$$

b) 拟合残差r_{mi}学习基学习器$T(x;\theta_m)$，训练集为$\{(x_i,r_{mi})\}_{i=1}^N$

c) 更新模型： $f_m(x)=f_{m-1}(x)+T(x;\theta_m)$

3) 得到最终的强学习器

$$f_M(x)=\sum_{m=1}^M T(x;\theta_m)$$

可以看出，提升树本质与Adboost一致，也是最小化第m个学习器和前$m-1$个学习器组合的损失函数，不同的是提升树采用决策树作为基学习器，采用残差的思想使得每个决策树的投票权重为1。

GBDT

GBDT是基学习器采用的Decision Tree的Gradient Boosting方法。Gradient Boosting模型与Adaboost的形式一致，采用M个基学习器的线性组合得到最终模型：

$$f_M(x)=\sum_m \gamma_m T(x;\theta_m)$$

首先确定初始模型，定义初始基学习器$f_0(x)$，当模型迭代到第m步时：

$$f_m(x)=f_{m-1}(x)+\gamma_m T(x;\theta_m)$$

通过最小化损失来确定参数θ_m的值：

$$arg\min_{\theta_m}\sum_i L(y_i,f_{m-1}(x_i)+\gamma_m T(x;\theta_m))$$

这里有两种理解Gradient Boosting的方式，从优化角度可以理解是采用梯度下降算法，T表示负梯度方向，γ_m为步长。从模型角度我们可以理解为损失函数一阶多项式展开$\gamma_m T(x,\theta_m)+f_{m-1}$，而$T$表示一阶信息，$\gamma_m$为系数。

优化角度，保证损失函数在递减：

$$L(y_i,f_m(x_i))<L(y_i,f_{m-1}(x_i))$$

为了使得损失函数不断减少，即梯度下降：

$$f_m(x_i) = f_{m-1}(x_i) + \gamma_m \left(-\frac{\partial L(y_i, f_{m-1}(x_i))}{\partial f_{m-1}(x_i)} \right)$$

$f_m(x) = f_{m-1}(x) + \gamma_m T(x;\theta_m)$代入上式有：

$$T(x;\theta_m) = -\frac{\partial L(y_i, f_{m-1}(x_i))}{\partial f_{m-1}(x_i)}$$

所以**Gradient Boosting** 的算法流程如下：

输入：训练集$\{(x_i, y_i)\}_{i=1}^N$，损失函数$L(y, f(x))$

输出：$f_M(x)$

1)初始化$f_0(x) = 0$

2)对$m = 1, 2, \dots M$

a) 计算梯度:

$$r_{mi} = \left[-\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f(x_i)=f_{m-1}(x)} \quad , \{1, 2, \dots, N\}$$

b) 拟合梯度r_{mi}学习基学习器$T(x;\theta_m)$，训练集为$\{(x_i, r_{mi})\}_{i=1}^N$

c) 根据梯度下降算法，计算学习器γ_m:

$$\gamma_m = arg\min_{\gamma} \sum_i L(y_i, f_{m-1}(x_i) + \gamma T(x;\theta_m))$$

d) 更新模型： $f_m(x) = f_{m-1}(x) + \gamma_m T(x;\theta_m)$

3) 得到最终的强学习器

$$f_M(x) = \sum_{m=1}^{M} \gamma_m T(x;\theta_m)$$

可以看出**Gradient Boosting** 是一个不断基于残差弥补的模型，目标不断地减少Bais，而没有关注Variance。它不像随机森林的集成引入随机性减少Variance的思想。

下面考虑决策树为基学习器的**Gradient Boosting**的方法GBDT，其在GB基础上有两点值得一提：

1) GBDT，采用决策树作为基函数将样本划分到固定数目J个决策区间$R_{mj}, j = 1, 2..J, m = 1, 2..M$

2) 在决策树中决策函数采用指示函数$I(x \in R_{mj})$，梯度与步长的积直接放到γ_{mj}上

下面给出GBDT回归和分类两个问题的算法流程

1) **GBDT 回归**

输入：训练集$\{(x_i, y_i)\}_{i=1}^N$，$x_i \in \mathbb{R}^n, y \in \mathbb{R}$，损失函数$L(y, f(x))$

输出：$f_M(x)$

1)初始时给出一个最优的偏置常数c，$f_0(x) = c$

$$f_0(x) = arg\min_{c} \sum_i L(y_i, c)$$

2)对$m=1,2,...M$

a) 计算梯度:

$$r_{mi}=\left[-\frac{\partial L(y_i,f(x_i))}{\partial f(x_i)}\right]_{f(x_i)=f_{m-1}(x)},\{1,2,...,N\}$$

b) 拟合梯度r_{mi}学习一个回归树$T(x;\theta_m)=\alpha I(x\in R_{mj})$，产生$J$个决策区间$R_{mj},j=1,2..J$

c) 对于决策区间$j=1,2..J$，计算γ_{mj}:

$$\gamma_{mj}=arg\min_{\gamma}\sum_{x_i\in R_{mj}}L(y_i,f_{m-1}(x_i)+\gamma_{mj}I(x\in R_{mj}))$$

d) 更新模型： $f_m(x)=f_{m-1}(x)+\sum_{j=1}^{J}\gamma_{mj}I(x\in R_{mj})$

3) 得到最终的强学习器

$$f_M(x)=\sum_{m=1}^{M}\sum_{j=1}^{J}\gamma_{mj}I(x\in R_{mj})$$

2) **GBDT分类**

考虑K分类问题，采用Softmax思想，将K类映射到K维。第m个决策树的决策第k维的值为$f_{M,k}(x)$，对输出进行Softmax归一化，可以得到k类的概率为$p_{m,k}(x)$，K类的概率和$\sum_k p_{m,k}(x)=1$，分类损失函数采用交叉熵损失。

$$p_{m,k}(x)=\frac{exp(f_{m,k}(x))}{\sum_{l=1}^{K}exp(f_{m,l}(x))},\quad k=1,...,K$$

似然函数为：

$$L(y_i,f_m(x_i))=\prod_{k=1}^{K}[f_{m,k}(x_i)]^{y_{ik}}$$

对数损失函数为：

$$L(y_i,f_m(x_i))=-\sum_{k=1}^{K}y_{ik}logf_{m,k}(x_i)$$

由于Softmax将分类映射到K维，对应的基分类器和损失函数都是K维。因此算法流程中负梯度方向也是一个K维向量。

输入：训练集$\{(x_i,y_i)\}_{i=1}^{N}$，$x_i\in\mathbb{R}^n,y\in\mathbb{R}$，损失函数$L(y,f(x))$

输出：$f_M(x)$

1)初始时$f_{0,k}(x)=0$

2)对$m=1,2,...M$

a) 对决策树$f_{m-1,k}$进行Softmax归一化

$$p_{m-1,k}(x)=\frac{exp(f_{m-1,k}(x))}{\sum_{l=1}^{K}exp(f_{m-1,l}((x))},\quad k=1,\ldots,K$$

b) 对$k=1,2..K$

ba) 计算梯度

$$r_{ik}=\frac{\partial L(y_i,f_{m-1}(x_i))}{\partial f_{m-1,k}(x_i)}=y_{ik}-p_{m-1,k}(x_i),\ \ i=1,2,\ldots N$$

bb) 拟合梯度r_{ik}学习第m个决策树$T(x;\theta_m)=\alpha I(x\in R_{mkj})$在第$k$维产生的$J$个决策区间$R_{mkj},j=1,2..J$

$$r_{ik}=\frac{\partial L(y_i,f_{m-1}(x_i))}{\partial f_{m-1,k}(x_i)}=y_{ik}-p_{m-1,k}(x_i),\ \ i=1,2,\ldots N$$

bc) 计算第m颗树第k维在区间R_{mj}的参数γ_{mkj}

$$\gamma_{mkj}=\frac{K-1}{K}\frac{\sum_{x_i\in R_{mkj}}r_{ik}}{\sum_{x_i\in R_{mkj}}|r_{ik}|(1-|r_{ik}|)},\quad j=1,2,\ldots,$$

bd) 更新模型： $f_{m,k}(x)=f_{m-1,k}(x)+\sum_{j=1}^{J}\gamma_{mkj}I(x\in R_{mkj})$

3) 得到最终的强学习器

$$f_M(x)=\sum_{m=1}^{M}\sum_{j=1}^{J}\gamma_{mj}I(x\in R_{mj})$$

GBDT到此，可以看出Boosting的这些方法，都是前向分步加法模型，分类回归采用不同的损失函数，加法模型都是考虑学习M模型来不断地减小损失函数，即第m个模型的学习是基于前$m-1$个模型组合起来最小化损失函数。Adboost是基于最小化损失函数在导数为0处取到，针对二分类问题导出样本系数，决策器权重系数，决策树。提升树是基于残差思想最小化损失函数，使得决策树的权重为1，GBDT采用一阶多项式来拟合残差，进而导出梯度提升的思想。GBDT中$\gamma_m T_m$存在冗余项，在GBDT中用决策树拟合梯度，γ来确定步长。

五、XGBoost

XGBoost是基于GBDT的一个改进，改进之处有以下几点：

1) 传统 GBDT以CART 作为基分类器，XGBoost 还支持线性分类器。

2) 目标函数中引入了正则项，决策树作为基学习器时约束定义叶子节点数和系数，线性分类器使用$L1$和$L2$

3) 损失函数的残差拟合使用二阶多项式拟合，不再采用梯度下降策略，而是导数为0导出决策树和权重参数

4) 支持并行，并行不是在基学习器层次，而是在特征选择层面，将特征列排序后存储，在迭代过程中重复使用

5) 样本采样，后剪枝处理防止过拟合

6) 对于缺失值处理，特征分裂时会默认给样本进行分类

XGBoost目标函数如下：

$$\min \sum_{i=1}^{N} L(y_i, \hat{y}_i^M) + \sum_{m=1}^{M} \Omega(f_m)$$
$$\text{where}: \Omega(f_m) = \gamma T + \frac{1}{2}||\omega||^2$$

在第m次迭代中有：

$$\min \sum_{i=1}^{N} L(y_i, \hat{y}_i^{m-1} + f_m(x_i)) + \sum_{m=1}^{M} \Omega(f_m)$$
$$\min \sum_{i=1}^{N} L(y_i, \hat{y}_i^{m-1} + \omega_{mi} f'_m(x_i) + \omega_{mi}^2 f''_m(x_i)) + \sum_{m=1}^{M} \Omega(f_m)$$

其中w_{mi}可以看做是损失函数在某一点的泰勒展开后的ΔX.

$g_{mi} = f'_m(x_i)$,$h_{mi} = f''_m(x_i)$，且采用决策树做基学习器，则有：

$$\min \sum_{i=1}^{N} L(y_i, \hat{y}_i^{m-1} + \omega_{mi} g_i + \omega_{mi}^2 h_i) + \gamma_m T_m + \lambda N ||\omega_{mi}||^2$$
$$= \min \sum_{i=1}^{N} L(y_i, \hat{y}_i^{m-1} + \omega_{mi} g_i + \omega_{mi}^2 (h_i + \lambda)) + \gamma_m T_m$$
$$= \min \sum_{i=1}^{N} L(y_i, \hat{y}_i^{m-1} + \omega_{mi} I\{i \in G_{mj}\} + \omega_{mi}^2 (I\{i \in H_{mj}\} + \lambda)) + \gamma_m T_m$$

其中w_{mi}是决策树的系数，G_{mi}是拟合一阶导的决策树，H_{mi}是拟合二阶导的决策树，而这两颗决策树可以共用决策树的结构，决策值不同构成一个二维输出的决策树。且二维决策树在两个维度的系数是有关系的,即w_{mi},w_{mi}^2.

对ω_j求导为0，有：

$$\omega_{mj}^* = -\frac{G_{mj}}{H_{mj} + \lambda}$$

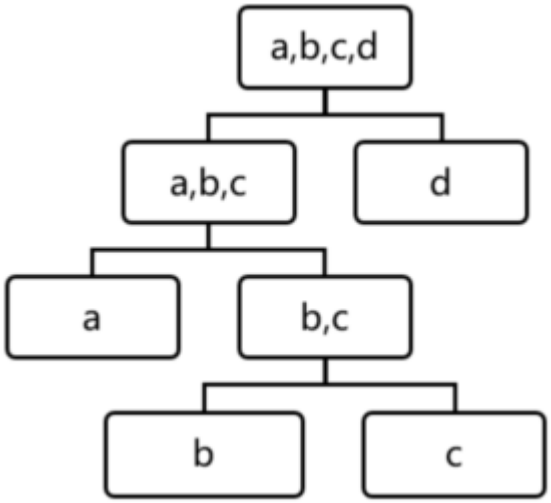
将ω_{mj}代入目标函数得：

$$Obj = -\frac{1}{2} \sum_{j=1}^{J} \frac{G_j^2}{H_j + \lambda} + \gamma_m T_m$$

由此，构建决策树的准则即为最小化目标函数，当决策树定下之后，ω也就定下。所以在XGBoost中实际上的参数只有决策树自身。

六、孤立森林

孤立森林（Isolation Foreset）是基于树（iTree）集成的快速异常检测方法，其异常检测的核心思想是“异常点是容易被孤立的离群点”。因此，孤立森林采用随机特征随机阈值划分生成多个树，直到树到达一定的高度或者直到每个叶子节点中只有一个点。那么，那些离群点很容易被提前（即所在叶子节点的深度较浅）被划分出来。由于每个树都是由随机采样独立生成的，所以树之间具有一定的独立性，多个树的集成就是最终的孤立森林。



可以看出，按照离群点大概率为异常点的话，那么d最有可能为异常点。

下面简单介绍孤立森林的流程和细节：

1）从训练集中随机选择（有放回和无放回）n个样本点构成子集$\Omega_i, i \in 1,2..m$，在m个子集上构建树

2）随机选择一个特征，随机选择一个阈值（最大值与最小值之间）进行二分裂

3）递归2）建树，直到树到达一定的高度d或者每个叶子节点中只有一个点。

4）m个树建好，根据m个决策树的平均深度来定义其异常的概率

a）统计每棵树的BST路径长度定义:

$$c(n) = 2H(n-1) - (2(n-1)/n)$$

b）定义异常的概率为：

$$s(x,n) = 2^{\left(-\frac{E(h(x))}{c(n)}\right)}$$

$c(n)$是$h(x)$在给定n下的平均值, 其中的$H(k)$可以通过公式 $H(k) = ln(k) + \xi$来估计，ξ是欧拉常数，其值为0.5772156649，k为从根节点到叶子节点的路径长度。

5）计算异常概率：

a）当$E(h(x)) = c(n)$，$s(x,n) = \frac{1}{2}$

b）当$E(h(x)) \to 0$，$s(x,n) = 1$

c）当$E(h(x)) \to n-1$，$s(x,n) = 0$

从上面建树的过程，可以看出孤立森林是针对连续值属性的，二分裂二叉树，当然离散值属性我想也是可以的。

孤立森林的特点：

1）每棵树随机采样独立生成，所以孤立森林具有很好的处理大数据的能力和速度

2）通常树的数量越多，算法越稳定，树的深度不易过深

3）孤立森林不适于特别高维的数据。因为子树的构建每次只选一个特征，容易受噪声影响

sklearn中孤立森林的参数设置：

n_estimators ：iTree的个数m

max_samples ：构建子树的样本数n，整数为个数，小数为占全集的比例

contamination ：异常的阈值

max_features ：构建每个子树的特征数，整数位个数，小数为占全特征的比例

bootstrap ：采样是有放回还是无放回

n_jobs ：并行数

random_state ：每次训练的随机性

verbose ：建树的过程描述输出