Part A: 决策树

1. 信息论基础

树具有天然的分支结构。对于分类问题而言，决策树的思想是用节点代表样本集合，通过某些判定条件来对节点内的样本进行分配，将它们划分到该节点下的子节点，并且要求各个子节点中类别的纯度之和应高于该节点中的类别纯度，从而起到分类效果。

节点纯度反映的是节点样本标签的不确定性。当一个节点的纯度较低时，说明每种类别都倾向于以比较均匀的频率出现，从而我们较难在这个节点上得到关于样本标签的具体信息，其不确定性较高。当一个节点的纯度很高时，说明有些类别倾向于以比较高的频率出现，从而我们能够更有信心地把握这个节点样本标签的具体信息，即确定性较高。

为了定义纯度的概念，我们首先需要思考如何度量不确定性。在生活中，高概率事件代表的不确定性比低概率事件代表的不确定性低，例如：明天太阳从东边升起是必然的，故这个事件的不确定性为0；而明天下雨并不是必然事件，它相比前一个事件具有更高的不确定性。因此，若定义一个可微函数$I(p), p \in [0,1]$来表示事件A发生的概率$p(A)$所代表的不确定性，那么从直觉上应当满足下面三个必要条件，我们将它们称为信息量公理。其中，$A_1, \ldots, A_n$为独立事件。

- $I(1) = 0$
- $I(p)$关于p单调递减
- $I(\prod_{i=1}^{n} p(A_i)) = \sum_{i=1}^{n} I(p(A_i))$

我们已经对信息量公理的前两个条件做出了说明，其第三个条件的含义是：独立事件同时发生的不确定性应当等于这些事件对应发生的不确定性之和，这是非常合理的假设。

根据这些条件，我们容易想到函数

$$I(p) = a\log_b(p)(a(b-1) < 0)$$

符合信息量公理的要求。事实上从充分性的角度而言，它也是能够满足信息量公理的唯一函数。

定理

设$I(p)$在$[0,1]$上可微，则$I(x)$满足信息量公理的充要条件是 $I(p) = a\log_b(p)(a(b-1) < 0)$。

证明

当$p \in (0,1]$，此时由导数定义有

$$\begin{aligned} I^{'}(p) &= \lim_{\Delta p \to 0^-} \frac{I(p+\Delta p) - I(p)}{\Delta p} \\ &= \lim_{\Delta p \to 0^-} \frac{I(\frac{p+\Delta p}{p} \cdot p) - I(p)}{\Delta p} \\ &= \lim_{\Delta p \to 0^-} \frac{I(\frac{p+\Delta p}{p})}{\Delta p} \\ &= \frac{1}{p} \lim_{\Delta p \to 0^-} \frac{I(1+\frac{\Delta p}{p})}{\frac{\Delta p}{p}} \\ &= \frac{1}{p} I^{'}(1) \end{aligned}$$

两边积分$I(p) = I^{'}(1)\ln p + C, p \in (0,1]$，代入$I(1) = 0$得$C = 0$，从而

$$I(p) = I^{'}(1)\ln(p) = \frac{I^{'}(1)}{\log_b e}\log_b p$$

记$a = \frac{I^{'}(1)}{\log_b e}$，由单调性可知$I^{'}(1) < 0$。当$b > 1$时有$\log_b e > 0$，即$a < 0$；当$b < 1$时有$\log_b e < 0$，即$a > 0$。因此，符合信息量公理的函数只能是$I(p) = a\log_b(p)(a(b-1) < 0)$。

我们已经知道了信息量对应函数的形式，那么究竟应该如何选取合适的a和b呢？对于一个以概率为p发生的事件A而言，我们可以选择一种二进制编码的方式来记录它的信息：当$p = \frac{1}{4}$时，我们可以认为事件A的发生本质上是某个随机变量的一种状态，且该随机变量会以等概率出现4种状态，那么我们就可以用00、01、10和11来进行状态信息的记录；当$p = \frac{1}{8}$时，我们需要用000、001、010、011、100、101、110、111的编码来记录。因此，p越小则不确定性越高，需要消耗的编码长度越大。此时，编码种类的数量即为$\frac{1}{p}$，事件A的二进制编码长度代表的不确定性大小就是$\log_2 \frac{1}{p}$。因此，我们可以取$I(p)$中的a为-1，且取b为2，用$I(p) = -\log_2(p)$来代表度量不确定性的指标。

先前我们讨论了随机变量取定某个值情况下不确定性的度量，那么如果要定义一个随机变量Y的平均不确定性，只需要对这个随机变量按照对应的概率密度分布$p(y)$取期望即可，我们将其称为分布的信息熵（Information Entropy）$H(Y)$（熵是一种反应系统不确定性的指标，由于此处指随机变量信息的不确定性，故称为信息熵），即

Contents

$$H(Y)=\mathbb{E}_Y I(p)=\mathbb{E}_{Y\sim p(y)}[-\log_2 p(Y)]$$

对于定义在有限状态集合$\{y_1,\dots,y_K\}$上的离散变量而言，对应信息熵的最大值在离散均匀分布时取到，最小值在单点分布时取到。此时，离散信息熵为

$$H(Y)=-\sum_{k=1}^{K}p(y_k)\log_2 p(y_k)$$

首先，我们需要定义当p时$p\log_2 p\triangleq 0$，原因在于

$$\lim_{p\to 0^+}p\log p=\lim_{p\to 0^+}\frac{\log p}{1/p}=\lim_{p\to 0^+}\frac{1/p}{-1/p^2}=\lim_{p\to 0^+}-p=0$$

离散熵的极值问题是带有约束的极值问题，记$p_k=P(Y=y_k)$和$\mathbf{p}=[p_1,\dots,p_K]^T$，则约束条件为$1^T\mathbf{p}=1$，拉格朗日函数为

$$L(\mathbf{p})=-\mathbf{p}^T\log\mathbf{p}+\lambda(1^T\mathbf{p}-1)$$

求偏导数后可解得$\mathbf{p}^*=[\frac{1}{K},\dots,\frac{1}{K}]$，此时$\mathbb{E}_Y I(p)=\log K$。

对于离散随机变量X，由于$p(Y)\in[0,1]$，故$-\log_2 p(Y)\geq 0$，从而$\mathbb{E}_Y I(p)\geq 0$。注意到对于$\forall k\in\{1,\dots,K\}$，当$p_k=1$，即$p_{k'}=0(k'\in\{1,\dots,K\}/k)$时，$H(X)=0$。因此，离散信息熵的最小值为0且在单点分布时取到。由于$\mathbf{p}^*$是极值问题的唯一解，因此离散熵的最大值为$\log K$且在离散均匀分布时取到。

这些结论都是与直觉高度吻合的。单点分布的取值被唯一确定，因此随机变量的不确定性为0；在给定状态集合数量下，分布越是均匀，则随机变量的不确定性越是大；当$K\to+\infty$时，离散均匀分布的熵为无穷大，一个合理的解释是：随着取值集合元素数量的增加，我们对每一个元素平均而言的信息把握程度就越少，不确定性就越大。

由于在决策树的分裂过程中，我们不但需要考察本节点的不确定性或纯度，而且还要考察子节点的平均不确定性或平均纯度来决定是否进行分裂。子节点的产生来源于决策树分支的条件，因此我们不但要研究随机变量的信息熵，还要研究在给定条件下随机变量的平均信息熵或条件熵（Conditional Entropy）。从名字上看，条件熵就是条件分布的不确定性，那么自然可以如下定义条件熵$H(Y|X)$为

$$\mathbb{E}_X[\mathbb{E}_{Y|X}[-\log_2 p(Y|X)]]$$

对于离散条件熵，设随机变量X所有可能的取值为$\{x_1,\dots,x_M\}$，上式可展开为

$$-\sum_{m=1}^{M}p(x_m)\sum_{k=1}^{K}p(y_k|X=x_m)\log_2 p(y_k|X=x_m)$$

有了信息熵和条件熵的基础，我们就能很自然地定义信息增益（Information Gain），即节点分裂之后带来了多少不确定性的降低或纯度的提高。在得到了随机变量X的取值信息时，随机变量Y不确定性的平均减少量为

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

从直觉上说，随机变量Y关于X的信息增益一定是非负的，因为我们额外地知道了随机变量X的取值，这个条件降低了Y的不确定性。下面我们就从数学角度来证明其正确性。

> **定理**
>
> 设Y和X为离散随机变量，Y关于X的信息增益$G(Y,X)$非负。

证明

由信息增益的定义得：

$$
\begin{aligned}
G(Y,X) &= \mathbb{E}_Y[-\log_2 p(Y)] - \mathbb{E}_X[\mathbb{E}_{Y|X}[-\log_2 p(Y|X)]] \\
&= -\sum_{k=1}^{K} p(y_k)\log_2 p(y_k) + \sum_{m=1}^{M} p(x_m)\sum_{k=1}^{K} p(y_k|X=x_m)\log_2 p(y_k|X=x_m) \\
&= -\sum_{k=1}^{K}[\sum_{m=1}^{M} p(y_k,x_m)]\log_2 p(y_k) + \sum_{k=1}^{K}\sum_{m=1}^{M} p(x_m)\frac{p(y_k,x_m)}{p(x_m)}\log_2\frac{p(y_k,x_m)}{p(x_m)} \\
&= \sum_{k=1}^{K}\sum_{m=1}^{M} p(y_k,x_m)[\log_2\frac{p(y_k,x_m)}{p(x_m)} - \log_2 p(y_k)] \\
&= -\sum_{k=1}^{K}\sum_{m=1}^{M} p(y_k,x_m)\log\frac{p(y_k)p(x_m)}{p(y_k,x_m)}
\end{aligned}
$$

从上式可以发现，信息增益$G(Y,X)$在本质上就是$p(y,x)$关于$p(y)p(x)$的KL散度，而KL散度的非负性由Jensen不等式可得：

$$
\begin{aligned}
G(X,Y) &\geq -\log_2[\sum_{k=1}^{K}\sum_{m=1}^{M} p(y_k,x_m)\frac{p(y_k)p(x_m)}{p(y_k,x_m)}] \\
&= -\log_2[\sum_{k=1}^{K}\sum_{m=1}^{M} p(y_k,x_m)] = 0
\end{aligned}
$$

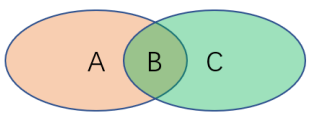
上述证明中的Jensen不等式的取等条件为$p(y,x)=p(y)p(x)$，其实际意义为随机变量Y和X独立。这个条件同样与直觉相符合，因为如果Y和X独立，那么意味着我们无论是否知道X的信息，都不会对Y的不确定性产生影响，此时信息增益为0。

用信息增益的大小来进行决策树的节点分裂时，由于真实的分布函数未知，故用$p(y)$和$p(y|x)$的经验分布（即频率）来进行概率的估计。若节点N每个分支下的样本数量为$D_1,\dots,D_M$，记$\tilde{p}(x_m)=\frac{D_m}{\sum_{m'=1}^{M}D_{m'}}(m\in\{1,\dots,M\})$，$\tilde{p}(y_k)$和$\tilde{p}(y_k|x_m)$分别为节点中第k个类别的样本占节点总样本的比例和第m个子节点中第k个类别的样本数量占该子节点总样本的比例，则节点N分裂的信息增益定义为

$$
G_N(Y,X) = -\sum_{i=1}^{K}\tilde{p}(y_k)\log\tilde{p}(y_k) + \sum_{m=1}^{M}\tilde{p}(x_m)\sum_{k=1}^{K}\tilde{p}(y_k|x_m)\log_2\tilde{p}(y_k|x_m)
$$

【练习】定义X,Y的联合熵为$H(Y,X)$为$\mathbb{E}_{(Y,X)\sim p(y,x)}[-\log_2 p(Y,X)]$

- 请证明如下关系：
  $G(Y,X)=H(X)-H(X|Y)$
  $G(Y,X)=H(X)+H(Y)-H(Y,X)$
  $G(Y,X)=H(Y,X)-H(X|Y)-H(Y|X)$
- 下图被分为了A、B和C三个区域。若AB区域代表X的不确定性，BC区域代表Y的不确定性，那么$H(X)$、$H(Y)$、$H(X|Y)$、$H(Y|X)$、$H(X,Y)$和$G(X,Y)$分别指代的是哪片区域？



2. 分类树的节点分裂

对于每个节点进行分裂决策时，我们会抽出max_features个特征进行遍历以比较信息增益的大小。特征的类别可以分为三种情况讨论：类别特征、数值特征和含缺失值的特征，它们各自的处理方法略有不同。

对于类别特征而言，给定一个阈值ϵ，树的每一个节点会选择最大信息增益$G_N^{max}(Y,X)$对应的特征进行分裂，直到所有节点的相对最大信息增益$\frac{D_N}{D_{all}}G_N^{max}(Y,X)$小于$\epsilon$，$D_N$和$D_{all}$分别指节点$N$的样本个数和整个数据集的样本个数，这种生成算法称为ID3算法。在sklearn中，ϵ即为min_impurity_decrease。

C4.5算法在ID3算法的基础上做出了诸多改进，包括但不限于：处理数值特征、处理含缺失值的特征、使用信息增益比代替信息增益以及给出树的剪枝策略。其中，剪枝策略将在第4节进行讲解，下面先对前3个改进的细节来进行介绍。

在处理节点数值特征时，可以用两种方法来将数值特征通过分割转化为类别，它们分别是最佳分割法和随机分割法，分别对应了sklearn中splitter参数的best选项和random选项。

随机分割法下，取$s\sim U[x_{min},x_{max}]$，其中$U[x_{min},x_{max}]$代表特征最小值和最大值范围上的均匀分布，将节点样本按照特征$\mathbf{x}$中的元素是否超过s把样本划分为两个集合，这等价于把数值变量转换为了类别变量。此时，根据这两个类别来计算树节点分裂的信息增益，并将它作为这个数值特征分裂的信息增益。

【练习】假设当前我们需要处理一个分类问题，请问对输入特征进行归一化会对树模型的类别输出产生影响吗？请解释原因。

最佳分割法下，依次令s取遍所有的$x_i(i=1,\dots,D_N)$，将其作为分割点，按照特征$\mathbf{x}$中的元素是否超过s把样本划分为两个集合，计算所有s对应信息增益的最大值，并将其作为这个数值特征分裂的信息增益。

C4.5算法处理缺失数据的思想非常简单，样本的缺失值占比越大，那么对信息增益的惩罚就越大，这是因为缺失值本身就是一种不确定性成分。设节点N的样本缺失值比例为γ，记非缺失值对应的类别标签和特征分别为$\tilde{Y}$和$\tilde{X}$，则修正的信息增益为

【练习】如果将系数替换为$1-\gamma^2$，请问对缺失值是加强了还是削弱了惩罚？

$$
\tilde{G}(Y,X) = (1-\gamma)G(\tilde{Y},\tilde{X})
$$

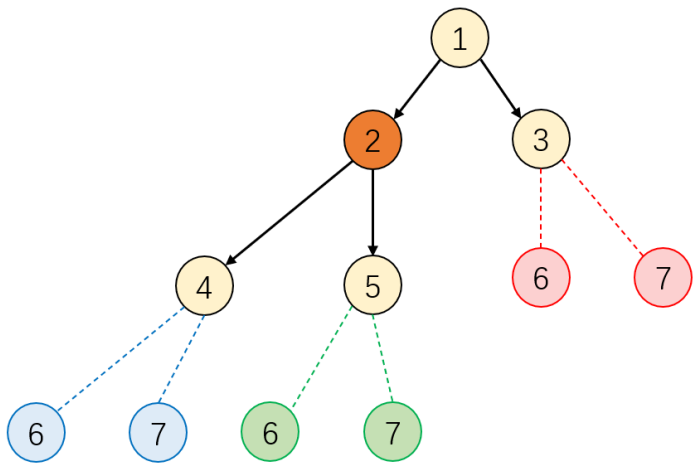
当数据完全缺失时$\gamma=1$，信息增益为0；当数据没有缺失值时$\gamma=0$，信息增益与原来的值保持一致。

在C4.5算法中，使用了信息增益比来代替信息增益，其原因在于信息增益来选择的决策树对类别较多的特征具有天然的倾向性，例如当某一个特征X（身份证号码、学号等）的类别数恰好就是样本数量时，此时由于$H(Y|X)=0$，即$G(Y,X)$达到最大值，因此必然会优先选择此特征进行分裂，但这样的情况是非常不合理的。

我们在第1节已经证明了，在类别占比均匀的情况下，类别数越多则熵越高，因此我们可以使用特征对应的熵来进行惩罚，即熵越高的变量会在信息增益上赋予更大程度的抑制，由此我们可以定义信息增益比为

$$G^R(Y,X)=\frac{G(Y,X)}{H(Y)}$$

在前面的部分中，我们讨论了单个节点如何选取特征进行分裂，但没有涉及到树节点的分裂顺序。例如下图所示，假设当前已经处理完了节点2的分裂，所有黄色节点（包括2号节点）都是当前已经存在的树节点，那么我们接下来究竟应该选取叶节点3号、4号和5号中的哪一个节点来继续进行决策以生成新的叶节点6号和7号？



在sklearn中提供了两种生长模式，它们分别被称为深度优先生长和最佳增益生长，当参数max_leaf_nodes使用默认值None时使用前者，当它被赋予某个数值时使用后者。

深度优先生长采用深度优先搜索的方法：若当前节点存在未搜索过的子节点，则当前节点跳转到子节点进行分裂决策；若当前节点为叶节点，则调转到上一层节点，直到根节点不存在未搜索过的子节点为止。对上图而言，当前节点为2号，它的两个子节点4号和5号都没有被搜索过，因此下一步则选择两个节点中的一个进行跳转。当决策树使用最佳增益生长时，每次总是选择会带来最大相对信息增益的节点进行分裂，直到叶节点的最大数量达到max_left_nodes。

【练习】如果将树的生长策略从深度优先生长改为广度优先生长，假设其他参数保持不变的情况下，两个模型对应的结果输出可能不同吗？

3. CART树

CART（Classification And Regression Tree）是一棵二叉树，它既能处理分类问题，又能够处理回归问题。值得注意的是，在sklearn中并没有实现处理类别特征和处理缺失值的功能，前者是因为多个类别的特征会产生多叉树，后者是因为sklearn认为用户应当自己决定缺失值的处理而不是交给模型来决定。

对于回归树而言，每个叶节点输出的不再是类别而是数值，其输出值为该叶节点所有样本标签值的均值。在每次分裂时，我们希望不同的子节点之间的差异较大，但每个子节点内部的差异较小。此时，分割策略仍然可以采用随机分割法或最佳分割法，只是现在不再以熵（条件熵）来评价节点（子节点）的纯度。

我们应当如何定义回归树的节点纯度？对于数值标签而言，我们可以认为节点间元素大小越接近则纯度越高，因此可以考虑使用均方误差（MSE）或平均绝对误差（MAE）来替换熵和条件熵的位置。

设节点N的样本标签为$y_1^{(D)},\dots,y_N^{(D)}$，左右子节点的样本个数分别为$y_1^{(L)},\dots,y_{N_R}^{(L)}$和$y_1^{(R)},\dots,y_{N_R}^{(R)}$，记$\bar{y}^{(D)}=\frac{1}{N}\sum_{i=1}^{N}y_i^{(D)}$、$\bar{y}^{(L)}=\frac{1}{N_L}\sum_{i=1}^{N_L}y_i^{(L)}$和$\bar{y}^{(R)}=\frac{1}{N_R}\sum_{i=1}^{N_R}y_i^{(R)}$分别为节点$N$的样本标签均值、左子节点的样本标签均值和右子节点的样本标签均值，再记$\tilde{y}^{(D)}$、$\tilde{y}^{(L)}$和$\tilde{y}^{(R)}$分别为节点N的样本标签中位数、左子节点的样本标签中位数和右子节点的样本标签中位数。

此时，两者的信息增益可以分别定义为

【练习】在一般的机器学习问题中，我们总是通过一组参数来定义模型的损失函数，并且在训练集上以最小化该损失函数为目标进行优化。请问对于决策树而言，模型优化的目标是什么？

$$G^{MSE}(Y,X)=\frac{1}{N}\sum_{i=1}^{N}(y_i^{(D)}-\bar{y}^{(D)})^2-\frac{N_L}{N}\frac{1}{N_L}\sum_{i=1}^{N_L}(y_i^{(L)}-\bar{y}^{(L)})^2-\frac{N_R}{N}\frac{1}{N_R}\sum_{i=1}^{N_R}(y_i^{(R)}-\bar{y}^{(R)})^2$$

$$G^{MAE}(Y,X)=\frac{1}{N}\sum_{i=1}^{N}|y_i^{(D)}-\tilde{y}^{(D)}|-\frac{N_L}{N}\frac{1}{N_L}\sum_{i=1}^{N_L}|y_i^{(L)}-\tilde{y}^{(L)}|-\frac{N_R}{N}\sum_{i=1}^{N_R}\frac{1}{N_R}|y_i^{(R)}-\tilde{y}^{(R)}|$$

当处理分类问题时，虽然ID3或C4.5定义的熵仍然可以使用，但是由于对数函数log的计算代价较大，CART将熵中的log在$p=1$处利用一阶泰勒展开，基尼系数定义为熵的线性近似，即由于

$$H(Y)=\mathbb{E}_Y I(p)=\mathbb{E}_Y[-\log_2 p(Y)]\approx\mathbb{E}_Y[1-p(Y)]$$

从而定义基尼系数为

$$\begin{aligned}\mathrm{Gini}(Y)&=\mathbb{E}_Y[1-p(Y)]\\&=\sum_{k=1}^{K}\tilde{p}(y_k)(1-\tilde{p}(y_k))\\&=1-\sum_{k=1}^{K}\tilde{p}^2(y_k)\end{aligned}$$

类似地定义条件基尼系数为

【练习】对信息熵中的log函数在$p=1$处进行一阶泰勒展开可以近似为基尼系数，那么如果在$p=1$处进行二阶泰勒展开我们可以获得什么近似指标？请写出对应指标的信息增益公式。

$$\begin{aligned}\text{Gini}(Y|X) &= \mathbb{E}_X[\mathbb{E}_{Y|X}1-p(Y|X)] \\ &= \sum_{m=1}^{M}\tilde{p}(x_m)\sum_{k=1}^{K}[\tilde{p}(y_k|x_m)(1-\tilde{p}(y_k|x_m))] \\ &= \sum_{m=1}^{M}\tilde{p}(x_m)[1-\sum_{k=1}^{K}\tilde{p}^2(y_k|x_m)]\end{aligned}$$

从而引出基于基尼系数的信息增益为

$$G(Y,X)=\text{Gini}(Y)-\text{Gini}(Y|X)$$

【练习】除了信息熵和基尼系数之外，我们还可以使用节点的$1-\max_k p(Y=y_k)$和第m个子节点的$1-\max_k p(Y=y_k|X=x_m)$来作为衡量纯度的指标。请解释其合理性并给出相应的信息增益和加权信息增益公式。

4. 决策树的剪枝

决策树具有很强的拟合能力，对于任何一个没有特征重复值的数据集，决策树一定能够在训练集上做到分类错误率或均方回归损失为0，因此我们应当通过一些手段来限制树的生长，这些方法被称为决策树树的剪枝方法。其中，预剪枝是指树在判断节点是否分裂的时候就预先通过一些规则来阻止其分裂，后剪枝是指在树的节点已经全部生长完成后，通过一些规则来摘除一些子树。

【练习】为什么对没有重复特征值的数据，决策树能够做到损失为0?

在sklearn的CART实现中，一共有6个控制预剪枝策略的参数，它们分别是最大树深度max_depth、节点分裂的最小样本数min_samples_split、叶节点最小样本数min_samples_leaf、节点样本权重和与所有样本权重和之比的最小比例min_weight_fraction_leaf、最大叶节点总数max_leaf_nodes以及之前提到的分裂阈值min_impurity_decrease。

【练习】如何理解min_samples_leaf参数能够控制回归树输出值的平滑程度?

后剪枝过程又称作MCCP过程，即Minimal Cost-Complexity Pruning，它由参数ccp_alpha控制，记其值为α。一般而言，树的叶子越多就越复杂，为了抑制树的生长，我们定义以节点N为根节点的树T^N的复杂度为该树的叶节点数量$|T^N|$。设树T的剪枝度量为

$$R_\alpha(T^N)=R(T^N)+\alpha|T^N|$$

其中，$R(T^N)$代表各个叶子节点的平均不纯度，此处的不纯度即指分类中的信息熵或者回归中的均方误差或平均绝对误差，即MCCP中的Cost部分，$\alpha|T^N|$对应的就是Complexity部分。

对于树的单个节点而言，由于此时节点数为1，故其剪枝度量为$R_\alpha(Node^N)=R(Node^N)+\alpha$。树剪枝的思想在于，如果对于决策树某一个节点为根的子树，其根的剪枝度量低于该子树的剪枝度量，那么这个根节点就没有必要分裂，即砍掉这棵子树中除了根节点以外的所有节点。

此时，我们可以得到剪枝的依据为

$$R_\alpha(Node^N)\leq R_\alpha(T^N)$$

这等价于

$$R(Node^N)+\alpha\leq R(T^N)+\alpha|T^N|$$

对上式进行移项后可得

$$E(Node^N)=\frac{R(Node^N)-R(T)}{|T^N|-1}\leq\alpha$$

这个条件表明只要$E(Node^N)$的值小于给定的参数cpp_alpha，那么这个节点下的所有节点都会被删除。事实上在sklearn中，在树完全生成后就会把所有节点的$E(Node^N)$值进行记录，每次剪枝都会分别查看所有非叶子节点的树节点对应的$E(Node^N)$值，并且对具有最小$E(Node^N)$值的非叶子节点进行剪枝，直到所有节点的$E(Node^N)$值都大于给定的cpp_alpha。

知识回顾

1. ID3树算法、C4.5树算法和CART算法之间有何异同?
2. 什么是信息增益？它衡量了什么指标？它有什么缺陷?
3. sklearn决策树中的random_state参数控制了哪些步骤的随机性?
4. 决策树如何处理连续变量和缺失变量?
5. 基尼系数是什么？为什么要在CART中引入它?
6. 什么是树的预剪枝和后剪枝？具体分别是如何操作的?

By GYH

Part B: 集成模式

1. 为何集成

我们在有限数据上训练模型，再用模型去预测新的数据，并期望在新数据上得到较低的预测损失，这里的预测损失可以指分类问题的错判率或回归问题的均方误差等各类评价指标。那预测数据产生的损失从何而来？这似乎并不是一个太好回答的问题，我们需要将上述的语境加以数学语言的规范。

对于实际问题中的数据，我们可以认为它总是由某一个分布p生成得到的，我们不妨设训练集合上有限的n个样本满足

$$\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n \sim p(\mathbf{X})$$

而这些样本对应的标签是通过真实模型f和噪声$\epsilon_1, \epsilon_2, \dots, \epsilon_n$得到的，即

$$y_i = f(\mathbf{X}_i) + \epsilon_i, i \in \{1, 2, \dots, n\}$$

其中，假设噪声均值为0且方差为σ^2。此时，我们得到了一个完整的训练集$D = \{(\mathbf{X}_1, y_1), (\mathbf{X}_2, y_2), \dots, (\mathbf{X}_n, y_n)\}$。对于新来的样本$\tilde{\mathbf{X}} \sim p(\mathbf{X})$，我们需要对其标签$y = f(\mathbf{X}_i) + \epsilon_i$进行预测，假设当前处理的是回归问题，应学习一个模型$\hat{f}$使得平方损失$(y - \hat{f}\,(\tilde{\mathbf{X}}))^2$尽可能地小。

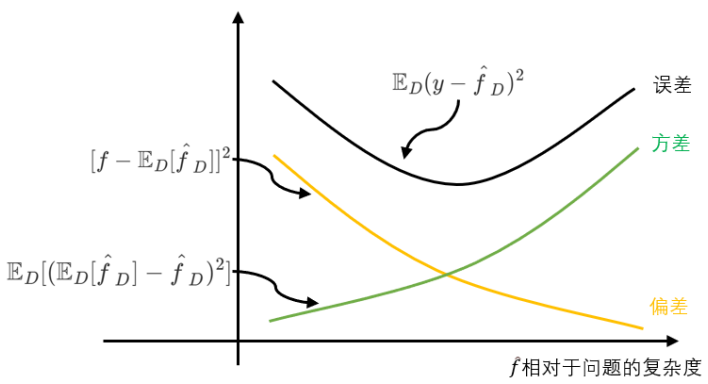
值得注意的是，$\hat{f}$不仅是$\tilde{\mathbf{X}}$的函数，由于它是从训练集上得到的，而训练集是从总体分布随机生成的有限分布，因此平方损失实际应记为为$(y - \hat{f}_{\,D}(\tilde{\mathbf{X}}))^2$。由于$D$是一个随机变量，故本质上优化的应当是$L = \mathbb{E}_D(y - \hat{f}_{\,D}(\tilde{\mathbf{X}}))^2$，这表示我们希望在按照给定分布任意生成的数据集上训练出的模型都能够有较好的预测能力，即模型具有良好的泛化性。

为了进一步研究L，我们把模型（基于不同数据集对新样本特征）的平均预测值信息$\mathbb{E}_D[\hat{f}_{\,D}(\tilde{\mathbf{X}})]$添加到$L$中，此时存在如下分解：

$$\begin{aligned}
L(\hat{f}\,) &= \mathbb{E}_D(y - \hat{f}_{\,D})^2 \\
&= \mathbb{E}_D(f + \epsilon - \hat{f}_{\,D} + \mathbb{E}_D[\hat{f}_{\,D}] - \mathbb{E}_D[\hat{f}_{\,D}])^2 \\
&= \mathbb{E}_D[(f - \mathbb{E}_D[\hat{f}_{\,D}]) + (\mathbb{E}_D[\hat{f}_{\,D}] - \hat{f}_{\,D}) + \epsilon]^2 \\
&= \mathbb{E}_D[(f - \mathbb{E}_D[\hat{f}_{\,D}])^2] + \mathbb{E}_D[(\mathbb{E}_D[\hat{f}_{\,D}] - \hat{f}_{\,D})^2] + \mathbb{E}_D[\epsilon^2] \\
&= [f - \mathbb{E}_D[\hat{f}_{\,D}]]^2 + \mathbb{E}_D[(\mathbb{E}_D[\hat{f}_{\,D}] - \hat{f}_{\,D})^2] + \sigma^2
\end{aligned}$$

上式中可见，预测数据产生的平均损失主要来自于三项。其中，第一项为数据真实值与模型平均预测值的偏差，偏差越小则代表模型的学习能力越强，能够较好地拟合数据，第二项为模型预测值的方差（我们要记住$\hat{f}$应视作随机变量D的函数，故$\hat{f}$是随机变量），方差越小则代表模型的抗干扰能力越强，不易因为数据的扰动而对预测结果造成大幅抖动，第三项为数据中的原始噪声，它是不可能通过优化模型来降解的。这种分解为我们设计模型提供了指导，即可以通过减小模型的偏差来降低测试数据的损失，也可以通过减少模型的预测方差来降低损失。

对于一个具体的学习问题，过于简单的学习器虽然抗干扰能力强，但由于不能充分拟合数据却会造成大的偏差，从而导致总的期望损失较高；类似地，过于复杂的学习器虽然其拟合能力很强，但由于学习了多余的局部信息，因此抗干扰能力较弱造成较大的方差，从而导致总的期望损失较高。上述联系由下图表达。



既然对单个学习器的偏差和方差的降解存在难度，那能否使用多个学习器进行结果的集成以进一步减小损失呢？更明确地说，我们希望利用多个低偏差的学习器进行集成来降低模型的方差，或者利用多个低方差学习器进行集成来降低模型的偏差，而bagging方法和boosting方法就分别是这两种思路的具体框架。

2. bagging与boosting

bagging是一种并行集成方法，其全称是$\mathbf{b}$ootstrap $\mathbf{ag}$gregat$\mathbf{ing}$，即基于bootstrap抽样的聚合算法，本章第4节中的随机森林算法就是一种基于bagging的模型。bootstrap抽样即指从样本集合中进行有放回的抽样，假设数据集的样本容量为n且基学习器的个数为M，对于每个基学习器我们可以进行有放回地抽取n个样本，从而生成了M组新的数据集，每个基学习器分别在这些数据集上进行训练，再将最终的结果汇总输出。

那这样的抽样方法有什么好处呢？我们已经知道数据集是从总体分布$p(\mathbf{X})$中抽样得到的，此时数据集构成了样本的经验分布$\tilde{p}(\mathbf{X})$，由于采用了不放回抽样，因此M个数据集中的每一组新样本都来自于经验分布$\tilde{p}(\mathbf{X})$。同时由大数定律知，当样本量$n \to \infty$时，经验分布$\tilde{p}(\mathbf{X})$收敛到总体分布$p(\mathbf{X})$，因此大样本下的新数据集近似地抽样自总体分布$p(\mathbf{X})$。

Contents

Print to PDF

假设我们处理的是回归任务，并且每个基学习器输出值$y^{(i)}$的方差为σ^2，基学习器两两之间的相关系数为ρ，则可以计算集成模型输出的方差为

$$
\begin{aligned}
Var(\hat{y}) &= Var(\frac{\sum_{i=1}^{M} y^{(i)}}{M}) \\
&= \frac{1}{M^2}[\sum_{i=1}^{M} Var(y^{(i)}) + \sum_{i \neq j} Cov(y^{(i)}, y^{(j)})] \\
&= \frac{1}{M^2}[M\sigma^2 + M(M-1)\rho\sigma^2] \\
&= \rho\sigma^2 + (1-\rho)\frac{\sigma^2}{M}
\end{aligned}
$$

从上式的结果来看，当基模型之间的相关系数为1时方差不变，这相当于模型之间的输出完全一致，必然不可能带来方差的降低。bootstrap的放回抽样特性保证了模型两两之间很可能有一些样本不会同时包含，这使模型的相关系数得以降低，而集成的方差随着模型相关性的降低而减小，如果想要进一步减少模型之间的相关性，那么就需要对基学习器进行进一步的设计。

为了更具体地了解bootstrap造成的数据集差异，我们往往对两个量是非常关心的，它们分别是单个样本的入选概率和入选（非重复）样本的期望个数。我们首先计算第一个量：对于样本容量为n的原数据集，抽到某一个给定样本的概率是$\frac{1}{n}$，进行n次bootstrap采样但却没有选入该样本的概率为$(1-\frac{1}{n})^n$，从而该样本入选数据集的概率为$1-(1-\frac{1}{n})^n$，当$n \to \infty$时的概率为$1-e^{-1}$。对于第二个量而言，记样本i在n次抽样中至少有一次入选这一事件为A_i，那么非重复样本的个数为$\sum_{i=1}^{n} 1_{\{A_i\}}$，从而期望个数为

$$
\begin{aligned}
\mathbb{E}\sum_{i=1}^{n} 1_{\{A_i\}} &= \sum_{i=1}^{n} \mathbb{E}1_{\{A_i\}} \\
&= \sum_{i=1}^{n} P(A_i) \\
&= \sum_{i=1}^{n} 1-(1-\frac{1}{n})^n \\
&= n[1-(1-\frac{1}{n})^n]
\end{aligned}
$$

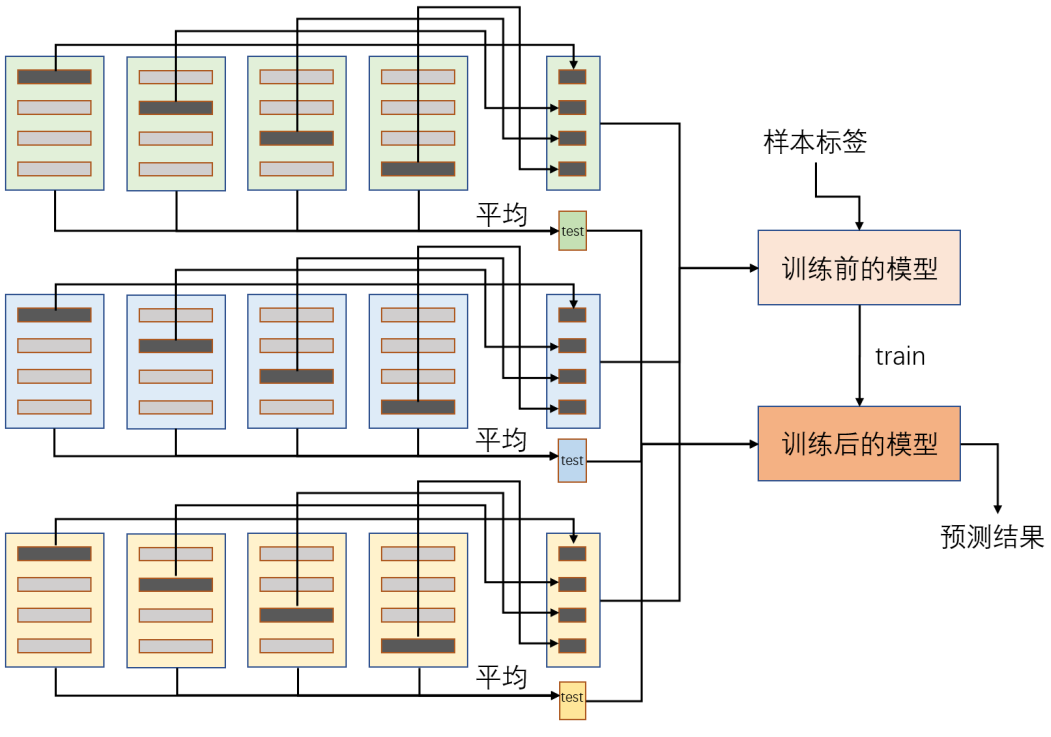
若$n \to \infty$时，入选样本占原数据集的期望比例为$\lim\limits_{n\to\infty} \dfrac{\mathbb{E}\sum_{i=1}^{n} 1_{\{A_i\}}}{n} = \lim\limits_{n\to\infty}[1-(1-\frac{1}{n})^n]$，即$1-\frac{1}{e}$。

boosting是一种串行集成方法，假设第i个基模型的输出是$\hat{f}^{(i)}(\mathbf{X})$，则总体模型的输出为$\sum_{i=1}^{M} \alpha_i \hat{f}^{(i)}(\tilde{X})$。boosting算法在拟合第$T$个学习器时，已经获得了前$T-1$个学习器的集成输出$\sum_{i=1}^{T-1} \alpha_i \hat{f}^{(i)}(\mathbf{X})$，对于损失函数$L(y, \hat{y})$，当前轮需要优化的目标即为使得$L(y, \alpha_T \hat{f}^{(T)}(\mathbf{X}) + \sum_{i=1}^{T-1} \alpha_i \hat{f}^{(i)}(\mathbf{X}))$最小化。需要强调的是，当前轮所有需要优化的参数一般而言都会蕴藏在$\alpha_T \hat{f}^{(T)}$这一项中，不同的模型会对$\alpha_T \hat{f}^{(T)}$提出的不同假设。此外，由于优化损失在经验分布与总体分布相差不多的时候等价于优化了模型的偏差，因此多个模型集成后相较于单个模型的预测能够使得偏差降低。我们将在后面的章节将进行具体模型的学习。

3. stacking与blending

stacking本质上也属于并行集成方法，但其并不通过抽样来构造数据集进行基模型训练，而是采用K折交叉验证。假设数据集大小为N，测试集大小为M，我们先将数据集均匀地分为K份。对于第m个基模型，我们取K折数据集中的1折为验证集，在剩下的$K-1$折为训练集，这样依次取遍K份验证数据就可以分别训练得到K个模型以及K份验证集上的相应预测结果，这些预测结果恰好能够拼接起来作为基模型在训练集上的学习特征F_m^{train}。同时，我们还需要利用这K个模型对测试集进行预测，并将结果进行平均作为该基模型的输出F_m^{test}。对每个基模型进行上述操作后，我们就能得到对应的训练集特征$F_1^{train}, \dots, F_M^{train}$以及测试集特征$F_1^{test}, \dots, F_M^{test}$。此时，我们使用一个最终模型$f$，以$F_1^{train}, \dots, F_M^{train}$为特征，以训练集的样本标签为目标进行训练。在$f$拟合完成后，以$F_1^{test}, \dots, F_M^{test}$为模型输入，得到的输出结果作为整个集成模型的输出。

整个stacking的流程如下图所示，我们在3种基学习器使用4折交叉验证，因此图中的左侧部分需要12次训练，右侧的浅红色和深红色表示的是同一个最终模型，使用不同颜色是为了主要区分训练前和训练后的状态。这里需要注意的是，整个集成模型一共包含了25次预测过程，即每个基模型对各折数据的预测、每个基模型对测试集数据的预测以及最终模型的1次预测。



blending集成与stacking过程类似，它的优势是模型的训练次数更少，但其缺陷在于不能使用全部的训练数据，相较于使用交叉验证的stacking稳健性较差。blending在数据集上按照一定比例划分出训练集和验证集，每个基模型在训练集上进行训练，并记验证集上的预测结果为$F_1^{train},\dots,F_M^{train}$。同时，我们还需要用这些基模型对测试集进行预测，其结果为$F_1^{test},\dots,F_M^{test}$。最后的流程与stacking一致，即以用$F_1^{train},\dots,F_M^{train}$和验证集的样本标签来训练最终模型，并将$F_1^{test},\dots,F_M^{test}$输入训练后的最终模型以得到模型的总体预测结果。假设仍然使用3种基学习器，由于无须交叉验证，此时只需要4次训练（含最终模型）和7次预测过程。

4. 两种并行集成的树模型

随机森林

随机森林是以决策树（常用CART树）为基学习器的bagging算法。当处理回归问题时，输出值为各学习器的均值；当处理分类问题时有两种策略，第一种是原始论文中使用的投票策略，即每个学习器输出一个类别，返回最高预测频率的类别，第二种是sklearn中采用的概率聚合策略，即通过各个学习器输出的概率分布先计算样本属于某个类别的平均概率，在对平均的概率分布取$\arg\max$以输出最可能的类别。

随机森林中的随机主要来自三个方面，其一为bootstrap抽样导致的训练集随机性，其二为每个节点随机选取特征子集进行不纯度计算的随机性，其三为当使用随机分割点选取时产生的随机性（此时的随机森林又被称为Extremely Randomized Trees）。

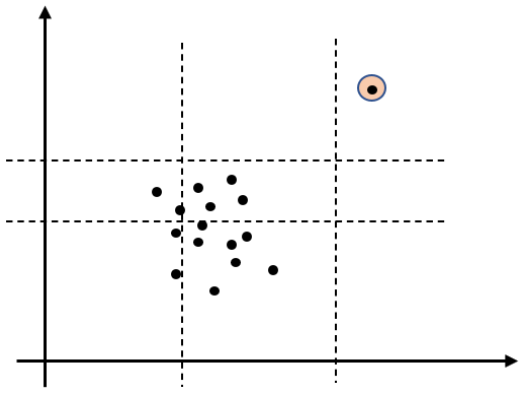
随机森林中特征重要性的计算方式为：利用相对信息增益来度量单棵树上的各特征特征重要性（与决策树计算方式一致），再通过对所有树产出的重要性得分进行简单平均来作为最终的特征重要性。

在训练时，一般而言我们总是需要对数据集进行训练集和验证集的划分，但随机森林由于每一个基学习器使用了重复抽样得到的数据集进行训练，因此总存在比例大约为$1-e^{-1}$的数据集没有参与训练，我们把这一部分数据称为out-of-bag样本，简称oob样本。此时，对每一个基学习器训练完毕后，我们都对oob样本进行预测，每个样本对应的oob_prediction_值为所有没有采样到该样本进行训练的基学习器预测结果均值，这一部分的逻辑参见此处的源码实现。在得到所有样本的oob_prediction_后，对于回归问题，使用r2_score来计算对应的oob_score_，而对于分类问题，直接使用accuracy_score来计算oob_score_。

最后，我们来介绍一种Totally Random Trees Embedding方法，它能够基于每个样本在各个决策树上的叶节点位置，得到一种基于森林的样本特征嵌入。具体地说，假设现在有4棵树且每棵树有4个叶子节点，我们依次对它们进行从0至15的编号，记样本i在4棵树叶子节点的位置编号为$[0,7,8,14]$，样本j的编号为$[1,7,9,13]$，此时这两个样本的嵌入向量即为$[1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0]$和$[0,1,0,0,0,0,0,1,0,1,0,0,0,1,0,0]$。假设样本$k$对应的编号为$[0,6,8,14]$，那么其对应嵌入向量的距离应当和样本$i$较近，而离样本$j$较远，即两个样本在不同树上分配到相同的叶子结点次数越多，则越接近。因此，这个方法巧妙地利用树结构获得了样本的隐式特征。

孤立森林

孤立森林也是一种使用树来进行集成的算法，其功能是用于连续特征数据的异常检测。孤立森林的基本思想是：多次随机选取特征和对应的分割点以分开空间中样本点，那么异常点很容易在较早的几次分割中就已经与其他样本隔开，正常点由于较为紧密故需要更多的分割次数才能将其分开。下图中体现了两个特征下的4次分割过程，可见右上角的异常点已经被单独隔离开。



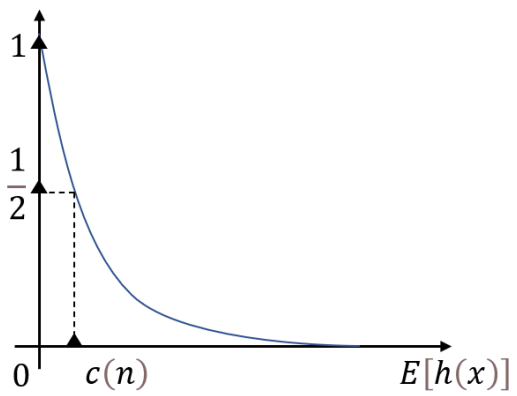
对于n个样本而言，我们可以构建一棵在每个分支进行特征大小判断的树来将样本分派到对应的叶子节点，为了定量刻画异常情况，在这篇文献中证明了树中的平均路径（即树的根节点到叶子结点经过的节点数）长度c为

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n}$$

其中$H(k)$为调和级数$\sum_{p=1}^{k} \frac{1}{p}$。此时对于某个样本$x$，假设其分派到叶子节点的路径长度为$h(x)$，我们就能用$\frac{h(x)}{c(n)}$的大小来度量异常的程度，该值越小则越有可能为异常点。由于单棵树上使用的是随机特征的随机分割点，稳健度较差，因此孤立森林将建立t棵树（默认100），每棵树上都在数据集上抽样出ψ个样本（默认256个）进行训练。为了总和集成的结果，我们定义指标

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

指数上的$\mathbb{E}h(x)$表示样本x在各树的路径平均值，当这个均值趋于0时，异常得分$s(x,n)$趋于1；当其趋于$n-1$时（n个样本最多需要$n-1$次分割，故树深度最大为$n-1$），$s(x,n)$趋于0（特别是在大样本情况下$c(n)$远小于$\mathbb{E}h(x)$）；当其趋于平均路径长度$\mathbb{E}h(x)$时，$s(x,n)$趋于$\frac{1}{2}$。变化关系如图所示。



虽然上述步骤明确了异常得分的计算，但是却还没有说明训练时树究竟应当在何时生长停止。事实上，我们可以规定树的生长停止当且仅当树的高度（路径的最大值）达到了给定的限定高度，或者叶子结点样本数仅为1，或者叶子节点样本数的所有特征值完全一致（即空间中的点重合，无法分离）。那么如何决定树的限定高度呢？在异常点判别的问题中，异常点往往是少部分的因此绝大多数的异常点都会在高度达到$c(n)$前被分配至路径较短的叶子结点，由于调和级数有如下关系（其中$\gamma \approx 0.5772$为欧拉常数）：

$$\lim_{n\to\infty} H(n) - \log n = \gamma$$

因此$c(n)$与$\log n$数量级相同，故给定的限定高度可以设置为$\log n$。此时，我们可以写出模型训练的伪代码（图片直接来自于原始论文）：

Algorithm 1 : $iForest(X, t, \psi)$

Inputs: X - input data, t - number of trees, ψ - sub-sampling size
Output: a set of t *iTrees*

Initialize $Forest$
set height limit $l = ceiling(\log_2 \psi)$
for $i = 1$ to t **do**
  $X' \leftarrow sample(X, \psi)$
  $Forest \leftarrow Forest \cup iTree(X', 0, l)$
end for
return $Forest$

Algorithm 2 : $iTree(X, e, l)$

Inputs: X - input data, e - current tree height, l - height limit
Output: an iTree

if $e \geq l$ or $|X| \leq 1$ **then**
  return *exNode*$\{Size \leftarrow |X|\}$
else
  let Q be a list of attributes in X
  randomly select an attribute $q \in Q$
  randomly select a split point p from max and min values of attribute q in X
  $X_l \leftarrow filter(X, q < p)$
  $X_r \leftarrow filter(X, q \geq p)$
  return *inNode*$\{Left \leftarrow iTree(X_l, e+1, l)$,
    $Right \leftarrow iTree(X_r, e+1, l)$,
    $SplitAtt \leftarrow q$,
    $SplitValue \leftarrow p\}$
end if

在预测阶段，应当计算样本在每棵树上被分配到叶子的路径，但我们还需要在路径长度上加上一项$c(T.size)$，其含义为当前叶子节点上样本数对应可生长的平均路径值，这是由于我们为了快速训练而对树高度进行了限制，事实上那些多个样本的节点仍然可以生长下去得到更长的路径，而新样本到达叶子结点后，平均而言还需要$c(T.size)$的路径才能达到真正（完全生长的树）的叶子结点。从而计算路径的伪代码如下图所示：

Algorithm 3 : $PathLength(x, T, e)$

Inputs : x - an instance, T - an iTree, e - current path length; to be initialized to zero when first called
Output: path length of x

1: **if** T is an external node **then**
2: return $e + c(T.size)$ $\{c(.)$ is defined in Equation 1$\}$
3: **end if**
4: $a \leftarrow T.splitAtt$
5: **if** $x_a < T.splitValue$ **then**
6: return $PathLength(x, T.left, e + 1)$
7: **else** $\{x_a \geq T.splitValue\}$
8: return $PathLength(x, T.right, e + 1)$
9: **end if**

在得到了各个树对应的路径后，我们就自然能计算$s(x,n)$了。假设我们需要得到前5%最可能为异常的点时，只需要对所有新样本的$s(x,n)$排序后输出前5%大的异常得分值对应的样本即可。

知识回顾

1. 什么是偏差和方差分解？偏差是谁的偏差？此处的方差又是指什么？
2. 相较于使用单个模型，bagging和boosting方法有何优势？
3. 请叙述stacking的集成流程，并指出blending方法和它的区别。
4. 什么是随机森林的oob得分？
5. 随机森林是如何集成多个决策树模型的？
6. 请叙述孤立森林的算法原理和流程。

By GYH

Part C: 自适应提升法

1. Adaboost概述

Adaboost的全称是Adaptive Boosting，其含义为自适应提升算法。其中，自适应是指Adaboost会根据本轮样本的误差结果来分配下一轮模型训练时样本在模型中的相对权重，即对错误的或偏差大的样本适度“重视”，对正确的或偏差小的样本适度“放松”，这里的“重视”和“放松”具体体现在了Adaboost的损失函数设计以及样本权重的更新策略。本课我们将介绍Adaboost处理分类和回归任务的算法原理，包括SAMME算法、SAMME.R算法和Adaboost.R2算法。

2. 分类损失

对于K分类问题而言，当样本标签$\mathbf{y}=[y_1,\ldots,y_K]^T$的类别$c$为第$k$类时，记

$$y_k=\begin{cases}1, & \text{if } c=k\\ -\dfrac{1}{k-1}, & \text{if } c\neq k\end{cases}$$

设模型的输出结果为$\mathbf{f}=[f_1,\ldots,f_K]^T$，则记损失函数为

$$L(\mathbf{y},\mathbf{f})=\exp(-\frac{\mathbf{y}^T\mathbf{f}}{K})$$

由于对任意的向量$a1$有

$$L(\mathbf{y},\mathbf{f}+a1)=\exp(-\frac{\mathbf{y}^T\mathbf{f}}{K}-\frac{a\mathbf{y}^T1}{K})=\exp(-\frac{\mathbf{y}^T\mathbf{f}}{K})=L(\mathbf{y},\mathbf{f})$$

因此为了保证$\mathbf{f}$的可估性，我们需要作出约束假设，此处选择对称约束条件

$$f_1+f_2+\ldots+f_K=0$$

从概率角度而言，一个设计良好的分类问题损失函数应当保证模型在期望损失达到最小时的输出结果是使得后验概率$P(c|\mathbf{x})$达到最大的类别，这个条件被称为贝叶斯最优决策条件。在本问题下，满足对称约束条件的损失函数期望损失$\mathbb{E}_{\mathbf{Y}|\mathbf{x}}L(\mathbf{Y},f)$达到最小时，由拉格朗日乘子法可解得模型输出为

$$\begin{aligned}k^*&=\arg\max_k f_k^*(\mathbf{x})\\&=\arg\max_k(K-1)[\log P(c=k|\mathbf{x})-\frac{1}{K}\sum_{i=1}^K\log P(c=i|\mathbf{x})]\\&=\arg\max_k P(c=k|\mathbf{x})\end{aligned}$$

因此，选择指数损失能够满足贝叶斯最优决策条件。

【练习】假设有一个3分类问题，标签类别为第2类，模型输出的类别标签为[-0.1,-0.3,0.4]，请计算对应的指数损失。

3. SAMME

SAMME算法的全称是$\mathbf{S}$tagewise $\mathbf{A}$dditive $\mathbf{M}$odeling using a $\mathbf{M}$ulticlass $\mathbf{E}$xponential loss function，它假定模型的总输出$\mathbf{f}$具有$\mathbf{f}^{(M)}(\mathbf{x})=\sum_{m=1}^M\beta^{(m)}\mathbf{b}^{(m)}(\mathbf{x})$的形式。其中，$M$是模型的总迭代轮数，$\beta^{(m)}\in\mathbb{R}^+$是每轮模型的加权系数，$\mathbf{b}^{(m)}(\mathbf{x})\in\mathbb{R}^K$是基模型$G$输出类别的标签向量。设样本的标签类别为$k$，当基模型预测的样本类别结果为$k'$时，记

$$b_{k'}^{(m)}=\begin{cases}1, & \text{if } k'=k\\ -\dfrac{1}{k-1}, & \text{if } k'\neq k\end{cases}$$

对于第m轮迭代而言，上一轮的模型输出为$\mathbf{f}^{(m-1)}(\mathbf{x})$，本轮需要优化得到的$\beta^{*(m)}$和$\mathbf{b}^{*(m)}$满足

$$(\beta^{*(m)},\mathbf{b}^{*(m)})=\arg\min_{\beta^{(m)},\mathbf{b}^{(m)}}\sum_{i=1}^n L(\mathbf{y}_i,\mathbf{f}^{(m-1)}(\mathbf{x}_i)+\beta^{(m)}\mathbf{b}^{(m)}(\mathbf{x}_i))$$

由于$\mathbf{f}^{(m-1)}(\mathbf{x}_i)$在第$m$轮为常数，记

$$w_i=\exp(-\frac{1}{K}\mathbf{y}_i^T\mathbf{f}^{(m-1)}(\mathbf{x}_i))$$

此时有

$$(\beta^{*(m)}, \mathbf{b}^{*(m)}) = \arg\min_{\beta^{(m)}, \mathbf{b}^{(m)}} \sum_{i=1}^{n} w_i \exp(-\frac{1}{K}\beta^{(m)} \mathbf{y}_i^T \mathbf{b}^{(m)}(\mathbf{x}_i))$$

设当轮预测正确的样本索引集合为T，则损失可表示为

$$\begin{aligned}
\tilde{L}(\beta^{(m)}, \mathbf{b}^{(m)}) &= \sum_{i=1}^{n} w_i \exp(-\frac{1}{K}\beta^{(m)} \mathbf{y}_i^T \mathbf{b}^{(m)}(\mathbf{x}_i)) \\
&= \sum_{i\in T} w_i \exp[-\frac{\beta^m}{K-1}] + \sum_{i\notin T} w_i \exp[\frac{\beta^{(m)}}{(K-1)^2}] \\
&= \sum_{i\in T} w_i \exp[-\frac{\beta^m}{K-1}] + \sum_{i\notin T} w_i \exp[-\frac{\beta^m}{K-1}] \\
&\quad - \sum_{i\notin T} w_i \exp[-\frac{\beta^m}{K-1}] + \sum_{i\notin T} w_i \exp[\frac{\beta^{(m)}}{(K-1)^2}] \\
&= \exp[-\frac{\beta^{(m)}}{K-1}] \sum_{i=1}^{n} w_i + \{\exp[\frac{\beta^{(m)}}{(K-1)^2}] - \exp[-\frac{\beta^{(m)}}{K-1}]\} \sum_{i=1}^{n} w_i \mathbb{I}_{\{i\notin T\}}
\end{aligned}$$

注意到$\mathbf{b}^{(m)}$仅与$\sum_{i=1} w_i \mathbb{I}_{\{i\notin T\}}$有关（因为基学习器的好坏控制了样本是否能够正确预测），且此项前的系数非负（因为$\beta^{(m)}$非负），因此得到

$$\mathbf{b}^{*(m)} = \arg\min_{\mathbf{b}^{(m)}} \sum_{i=1}^{n} w_i \mathbb{I}_{\{i\notin T\}}$$

在得到$\mathbf{b}^{*(m)}$后，通过求$\tilde{L}$关于$\beta^{(m)}$的导数并令之为0可解得

$$\beta^{*(m)} = \frac{(K-1)^2}{K}[\log\frac{1-err^{(m)}}{err^{(m)}} + \log(K-1)]$$

其中，样本的加权错误率为

$$err^{(m)} = \sum_{i=1}^{n} \frac{w_i}{\sum_{i=1}^{n} w_i} \mathbb{I}_{\{i\notin T\}}$$

样本$\mathbf{x}_i$在第m轮的预测类别为$k_i^* = \arg\max_k \mathbf{f}^{(m)}(\mathbf{x}_i)$，其中

$$\mathbf{f}^{(m)}(\mathbf{x}_i) = \mathbf{f}^{(m-1)}(\mathbf{x}_i) + \beta^{*(m)} \mathbf{b}^{*(m)}(\mathbf{x}_i)$$

将上述算法过程总结伪代码如下：

【练习】对公式进行化简，写出$K=2$时的SAMME算法流程，并与李航《统计学习方法》一书中所述的Adaboost二分类算法对比是否一致。

Algorithm 1: Adaboost 方法的 SAMME 实现

Data: 训练样本 $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_n)$ 和 $\mathbf{y} = (\mathbf{y}_1, ..., \mathbf{y}_n)$、基分类器 G、迭代轮数 M、测试样本 $\mathbf{x}$
Result: 测试样本的预测类别 $c(\mathbf{x})$

1 **for** $i \leftarrow 1$ **to** n **do**
2 $\quad w_i \leftarrow \frac{1}{n}$
3 **end**
4 **for** $m \leftarrow 1$ **to** M **do**
5 $\quad G^* \leftarrow \arg\min_G \sum_{i=1}^{n} w_i \mathbb{I}_{\{i\notin T\}}$
6 $\quad err^{(m)} \leftarrow \sum_{i=1}^{n} \frac{w_i}{\sum_{i=1}^{n} w_i} \mathbb{I}_{\{i\notin T\}}$
7 $\quad \beta^{*(m)} \leftarrow \frac{(K-1)^2}{K}[\log\frac{1-err^{(m)}}{err^{(m)}} + \log(K-1)]$
8 $\quad$ **for** $i \leftarrow 1$ **to** n **do**
9 $\quad\quad \mathbf{b}^{*(m)}(\mathbf{x}_i) \leftarrow G^*(\mathbf{x}_i)$
10 $\quad\quad w_i \leftarrow w_i \cdot \exp(-\frac{1}{K}\beta^{*(m)} \mathbf{y}_i^T \mathbf{b}^{*(m)}(\mathbf{x}_i))$
11 $\quad$ **end**
12 $\quad \mathbf{f}^{(m)} \leftarrow \mathbf{f}^{(m-1)} + \beta^{*(m)} \mathbf{b}^{*(m)}$
13 **end**
14 $c(\mathbf{x}) \leftarrow \arg\max_k \mathbf{f}^{(M)}(\mathbf{x})$

事实上，我们还能通过一些多分类的性质来改写算法的局部实现，使得一些变量前的系数得到简化。记

$$\alpha^{*(m)} = \log\frac{1-err^{(m)}}{err^{(m)}} + \log(K-1)$$

此时，w_i每轮会被更新为

$$\tilde{w}_i = w_i \cdot \exp[\frac{1-K}{K}\alpha^{*(m)}] \exp(\alpha^{*(m)} 1_{\{i\notin T\}})$$

对$\mathbf{w}$进行归一化操作后，不会对下一轮算法1中G^*和$err^{(m)}$的结果产生任何影响。同时，如果把算法1第12行的$\beta^{*(m)}$替换为$\alpha^{*(m)}$，由于它们的输出结果只相差常数倍$\frac{(K-1)^2}{K}$，因此最后的预测结果$c(\mathbf{x})$也不会产生任何变化。

【练习】在sklearn源码中找出算法流程中每一行对应的处理代码。

由于$\exp[\frac{1-K}{K}\alpha^{*(m)}]$是样本公共项，故我们可以每次都利用

$$\tilde{w}_i = w_i \cdot \exp(\alpha^{*(m)} 1_{\{i\notin T\}})$$

【练习】算法2第12行中给出了$\mathbf{f}$输出的迭代方案，但在sklearn包的实现中使用了$\mathbb{I}_{\{G^*(\mathbf{x})=S(\mathbf{y})\}}$来代替$\mathbf{b}^{*(m)}(\mathbf{x})$。请根据本文的实现，对sklearn包的源码进行修改并构造一个例子来比较它们的输出是否会不同。

来更新，而不影响归一化结果。此时，算法1的迭代循环可进行如下重写：

（提示：修改AdaboostClassifier类中的decision_function函数和staged_decision_function函数）

Algorithm 2: SAMME 算法迭代循环的优化实现

1 **for** $m \leftarrow 1$ **to** M **do**
2 $\quad G^* \leftarrow \arg\min_G \sum_{i=1}^n w_i \mathbb{I}_{\{i \notin T\}}$
3 $\quad err^{(m)} \leftarrow \sum_{i=1}^n w_i \mathbb{I}_{\{i \notin T\}}$
4 $\quad \alpha^{*(m)} \leftarrow \log \frac{1-err^{(m)}}{err^{(m)}} + \log(K-1)$
5 $\quad$ **for** $i \leftarrow 1$ **to** n **do**
6 $\quad\quad \mathbf{b}^{*(m)}(\mathbf{x}_i) \leftarrow G^*(\mathbf{x}_i)$
7 $\quad\quad \tilde{w}_i \leftarrow w_i \cdot \exp(\alpha^{*(m)} \mathbb{1}_{\{i \notin T\}})$
8 $\quad$ **end**
9 $\quad$ **for** $i \leftarrow 1$ **to** n **do**
10 $\quad\quad w_i \leftarrow \frac{w_i}{\sum_{i=1}^n w_i}$
11 $\quad$ **end**
12 $\quad \mathbf{f}^{(m)} \leftarrow \mathbf{f}^{(m-1)} + \alpha^{*(m)} \mathbf{b}^{*(m)}$
13 **end**

4. SAMME.R

许多分类器都能够输出预测样本所属某一类别的概率，但是SAMME算法只能利用分类的标签信息，而不能利用这样的概率信息。SAMME.R算法通过损失近似的思想，将加权分类模型的概率输出信息与boosting方法相结合。SAMME.R中的字母“R”代表“Real”，意味着模型每轮迭代的输出为实数。

不同于SAMME在第m轮需要同时考虑得到最优的$\beta^{(m)}$和$\mathbf{b}^{(m)}$，SAMME.R将其统一为$\mathbf{h}^{(m)} \in \mathbb{R}^K$，它需要满足对称约束条件$\sum_{i=1}^K h_k = 0$以保证可估性。此时，损失函数为

$$L(\mathbf{h}^{(m)}) = \exp[-\frac{1}{K}\mathbf{y}^T(\mathbf{f}^{(m-1)}(\mathbf{x}) + \mathbf{h}^{(m)}(\mathbf{x}))]$$

为了与概率联系，我们需对损失L的后验概率进行最小化，即

$$\begin{aligned}\mathbf{h}^{*(m)} &= \arg\min_{\mathbf{h}^{(m)}} \mathbb{E}[L|\mathbf{x}] \\ &= \arg\min_{\mathbf{h}^{(m)}} \mathbb{E}_{\mathbf{y}}[\exp[-\frac{1}{K}\mathbf{y}^T(\mathbf{f}^{(m-1)}(\mathbf{x}) + \mathbf{h}^{(m)}(\mathbf{x}))]|\mathbf{x}]\end{aligned}$$

设样本$\mathbf{y}$对应的标签为$S(\mathbf{y})$，则

$$\begin{aligned}\mathbb{E}[L|\mathbf{x}] &= \mathbb{E}_{\mathbf{y}}[\exp[-\frac{1}{K}\mathbf{y}^T\mathbf{f}^{(m-1)}(\mathbf{x})]\exp[-\frac{1}{K}\mathbf{y}^T\mathbf{h}^{(m)}(\mathbf{x})]]|\mathbf{x}] \\ &= \sum_{k=1}^K [\exp[-\frac{1}{K}\mathbf{y}^T\mathbf{f}^{(m-1)}(\mathbf{x})]\exp[-\frac{1}{K}\mathbf{y}^T\mathbf{h}^{(m)}(\mathbf{x})]]\Big|_{S(\mathbf{y})=k} P(S(\mathbf{y})=k|\mathbf{x}) \\ &= \sum_{k=1}^K [\exp[-\frac{1}{K}\mathbf{y}^T\mathbf{f}^{(m-1)}(\mathbf{x})]\Big|_{S(\mathbf{y})=k} P(S(\mathbf{y})=k|\mathbf{x})]\exp(-\frac{h_k^{(m)}(\mathbf{x})}{K-1})\end{aligned}$$

记$w = \exp[-\frac{1}{K}\mathbf{y}^T\mathbf{f}^{(m-1)}(\mathbf{x})]$，则

$$\mathbb{E}[L|\mathbf{x}] = \sum_{k=1}^K w|_{S(\mathbf{y})=k} \cdot P(S(\mathbf{y})=k)\exp(-\frac{h_k^{(m)}(\mathbf{x})}{K-1})$$

不难发现对于样本$\mathbf{y}$而言，越大的w意味着上一轮的模型结果越糟糕，此时负责预测$P(S(\mathbf{y})=k)$的基模型就要加大对该样本的重视程度以获得较小的损失。

但是，此时基模型本身是不带权重的，SAMME.R采用的近似方法是，考虑以w为权重的基模型G，用其输出$P_w(s(\mathbf{y})=k|\mathbf{x})$的概率值来代替$w|_{S(\mathbf{y})=k} \cdot P(S(\mathbf{y})=k|\mathbf{x})$，这种行为合法的原因在于权重对于总体损失的惩罚方向是一致的，G通过权重w将原本作用于L的损失近似地“分配”给了基分类器的损失。

此时，损失函数近似为

$$\mathbb{E}[L|\mathbf{x}] = \sum_{k=1}^K P_w(s(\mathbf{y})=k|\mathbf{x})\exp(-\frac{h_k^{(m)}(\mathbf{x})}{K-1})$$

由对称约束条件，结合拉格朗日乘子法可得

【练习】验证$h_{k'}^{*}$的求解结果。

$$h_{k'}^{*(m)} = (K-1)[\log P_w(S(\mathbf{y})=k'|\mathbf{x}) - \frac{1}{K}\sum_{k=1}^K \log P(S(\mathbf{y})=k|\mathbf{x})]$$

将上述算法过程总结伪代码如下：

【练习】算法3的第14行给出了w_i的更新策略，请说明其合理性。

Algorithm 3: Adaboost 方法的 SAMME.R 实现（输入和输出同 SAMME）

for $i \leftarrow 1$ **to** n **do**
  $w_i \leftarrow \frac{1}{n}$
end
for $m \leftarrow 1$ **to** M **do**
  $G^* \leftarrow$ 以 $\mathbf{w}$ 为权重训练的基模型
  for $i \leftarrow 1$ **to** n **do**
    for $k \leftarrow 1$ **to** K **do**
      $P_k^{(m)}(\mathbf{x}_i) \leftarrow P_w(S(\mathbf{y}_i) = k|\mathbf{x})$
    end
  end
  for $i \leftarrow 1$ **to** n **do**
    for $k' \leftarrow 1$ **to** K **do**
      $h_{k'}^{(m)}(\mathbf{x}_i) \leftarrow (K-1)[\log P_{k'}^{(m)}(\mathbf{x}) - \frac{1}{K}\sum_{k=1}^{K} \log P_k^{(m)}((x))]$
      $w_i \leftarrow w_i \cdot \exp(-\frac{K-1}{K}\mathbf{y}_i^T[\log P_1^{(m)}, ..., \log P_K^{(m)}])$
    end
  end
  for $i \leftarrow 1$ **to** n **do**
    $w_i \leftarrow \frac{w_i}{\sum_{i=1}^{n} w_i}$
  end
end
$c(\mathbf{x}) \leftarrow \arg\max_k \sum_{m=1}^{M} h_k^{(m)}(\mathbf{x})$

5. Adaboost.R2

利用权重重分配的思想，Adaboost还可以应用于处理回归问题。其中，Adaboost.R2算法是一种最常使用的实现。

设训练集特征和目标分别为$\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$和$\mathbf{y} = (y_1, \ldots, y_n)$，权重$\mathbf{w}$初始化为$(w_1, \ldots, w_n)$。在第$m$轮时，根据权重训练基预测器得到$G^*$，计算每个样本的相对误差

$$e_i = \frac{|y_i - G^*(\mathbf{x}_i)|}{\max_i |y_i - G^*(\mathbf{x}_i)|}$$

设样本的加权相对误差率为$E^{(m)} = \sum_{i=1}^{n} w_i e_i$，则相对误差率与正确率的比值为$\beta^{(m)} = \frac{E^{(m)}}{1-E^{(m)}}$，即预测器权重$\alpha^{(m)} = \log \frac{1}{\beta^{(m)}}$。

更新权重w_i为$w_i[\alpha^{(m)}]^{1-e_i}$，权重在归一化后进入下一轮训练，由此可如下写出训练算法：

Algorithm 4: Adaboost.R2 算法的训练流程

for $i \leftarrow 1$ **to** n **do**
  $w_i \leftarrow \frac{1}{n}$
end
for $m \leftarrow 1$ **to** M **do**
  for $i \leftarrow 1$ **to** n **do**
    $e_i \leftarrow \frac{|y_i - G^*(\mathbf{x}_i)|}{\max_i |y_i - G^*(\mathbf{x}_i)|}$
  end
  $E^{(m)} \leftarrow \sum_{i=1}^{n} w_i e_i$
  $\beta^{(m)} \leftarrow \frac{E^{(m)}}{1-E^{(m)}}$
  $\alpha^{(m)} \leftarrow \log \frac{1}{\beta^{(m)}}$
  $w_i \leftarrow w_i[\alpha^{(m)}]^{1-e_i}$
  for $i \leftarrow 1$ **to** n **do**
    $w_i \leftarrow \frac{w_i}{\sum_{i=1}^{n} w_i}$
  end
end

在预测阶段，Adaboost.R2使用的是加权中位数算法。设每个基模型对某一个新测试样本的预测输出为$y_1, \ldots, y_M$，基模型对应的预测器权重为$\alpha^{(1)}, \ldots, \alpha^{(M)}$，则Adaboost.R2的输出值为

$$y = \inf\{y | \sum_{m \in \{m | y_m \leq y\}} \alpha^{(m)} \geq 0.5 \sum_{m=1}^{M} \alpha^{(m)}\}$$

【练习】请结合加权中位数的定义解决以下问题：

- 当满足什么条件时，Adaboost.R2的输出结果恰为每个基预测器输出值的中位数？
- Adaboost.R2模型对测试样本的预测输出值是否一定会属于M个分类器中的一个输出结果？若是请说明理由，若不一定请给出反例。
- 设$k \in \{y_1, \ldots, y_M\}$，记$k$两侧（即大于或小于$k$）的样本集合对应的权重集合为$W^+$和$W^-$，证明使这两个集合元素之和差值最小的$k$就是Adaboost.R2输出的$y$。
- 相对于普通中位数，加权中位数的输出结果鲁棒性更强，请结合公式说明理由。

知识回顾

1. 二分类问题下，Adaboost算法如何调节样本的权重？
2. 样本A在当轮分类错误，且样本B在当轮分类正确，请问在权重调整后，样本A的权重一定大于样本B吗？
3. 在处理分类问题时，Adaboost的损失函数是什么？请叙述其设计的合理性。
4. Adaboost如何处理回归问题？
5. 用已经训练完毕的Adaboost分类模型和回归的模型来预测新样本的标签，请分别具体描述样本从输入到标签输出的流程。

By GYH

Part D: 梯度提升树

1. 用于回归的GBDT

设数据集为$D=\{(X_1,y_1),\dots,(X_N,y_N)\}$，模型的损失函数为$L(y,\hat{y})$，现希望利用多棵回归决策树来进行模型集成：设第$m$轮时，已知前$m-1$轮中对第$i$个样本的集成输出为$F_{m-1}(X_i)$，则本轮的集成输出$\hat{y}_i$为

$$F_m(X_i)=F_{m-1}(X_i)+h_m(X_i)$$

其中，h_m是使得当前轮损失$\sum_{i=1}^{N}L(y_i,\hat{y}_i)$达到最小的决策树模型。

特别地，当$m=0$时，$F_0(X_i)=\arg\min_{\hat{y}}\sum_{i=1}^{N}L(y_i,\hat{y})$。

记第m轮的损失函数为

$$G(h_m)=\sum_{i=1}^{N}L(y_i,F_{m-1}(X_i)+h_m(X_i))$$

令上述损失最小化不同于一般的参数优化问题，我们需要优化的并不是某一组参数，而是要在所有决策树模型组成的函数空间中，找到一个h^*使得$G(h^*)$最小。因此我们不妨这样思考：学习一个决策树模型等价于对数据集$\tilde{D}=\{(X_1,h^*(X_1)),\dots,(X_N,h^*(X_N))\}$进行拟合，设$w_i=h^*(X_I)$，$\mathbf{w}=[w_1,\dots,w_N]$，此时的损失函数可改记为

$$G(\mathbf{w})=\sum_{i=1}^{N}L(y_i,F_{m-1}(X_i)+w_i)$$

由于只要我们获得最优的$\mathbf{w}$，就能拟合出第m轮相应的回归树，此时一个函数空间的优化问题已经被转换为了参数空间的优化问题，即对于样本i而言，最优参数为

$$w_i=\arg\min_{w}L(y_i,F_{m-1}(X_i)+w)$$

对于可微的损失函数L，由于当$\mathbf{w}=\mathbf{0}$时的损失就是第$m-1$轮预测产生的损失，因此我们只需要在$w_i=0$处进行一步梯度下降（若能保证合适的学习率大小）就能够获得使损失更小的w_i^*，而这个值正是我们决策树需要拟合的$h^*(X_i)$。 以损失函数$L(y,\hat{y})=\sqrt{|y-\hat{y}|}$为例，记残差为

$$r_i=y_i-F_{m-1}(X_i)$$

则实际损失为

$$L(w_i)=\sqrt{|r_i-w_i|}$$

根据在零点处的梯度下降可知：

$$\begin{aligned}w_i^*&=0-\frac{\partial L}{\partial w}\Big|_{w=0}\\&=-\frac{1}{2\sqrt{r_i}}sign(r_i)\end{aligned}$$

为了缓解模型的过拟合现象，我们需要引入学习率参数η来控制每轮的学习速度，即获得了由$\mathbf{w}^*$拟合的第m棵树h^*后，当前轮的输出结果为

$$\hat{y}_i=F_{m-1}(X_i)+\eta h_m^*(X_i)$$

对于上述的梯度下降过程，还可以从另一个等价的角度来观察：若设当前轮模型预测的输出值为$\tilde{w}_i=F_{m-1}(X_i)+w_i$，求解的问题即为

$$\tilde{w}_i=\arg\min_{\tilde{w}}L(y_i,\tilde{w})$$

由于当$\tilde{w}=F_{m-1}(X_i)$时，损失函数的值就是上一轮预测结果的损失值，因此只需将L在$\tilde{w}$在$\tilde{w}=F_{m-1}(X_i)$的位置进行梯度下降，此时当前轮的预测值应为

$$\tilde{w}_i^*=F_{m-1}(X_i)-\frac{\partial L}{\partial \tilde{w}}\Big|_{\tilde{w}=F_{m-1}(X_i)}$$

从而当前轮学习器h需要拟合的目标值w_i^*为

Contents

Print to PDF

【练习】对于均方损失函数和绝对值损失函数，请分别求出模型的初始预测F_0。

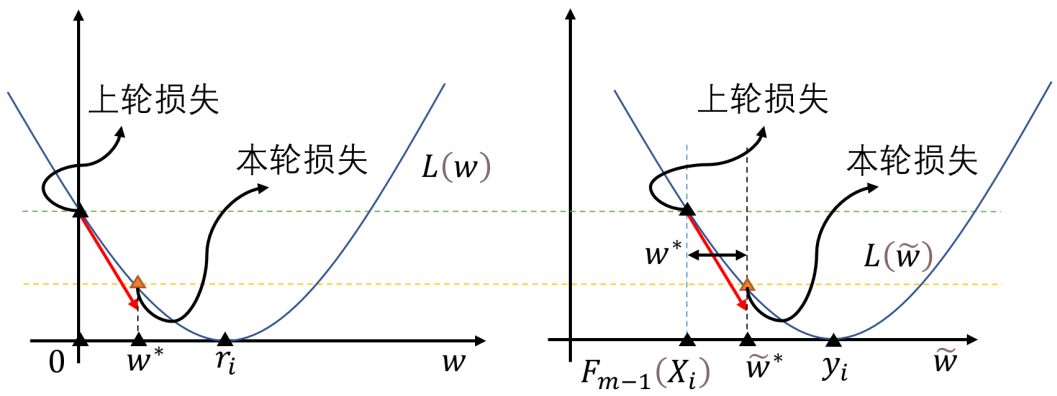
【练习】给定了上一轮的预测结果$F_{m-1}(X_i)$和样本标签y_i，请计算使用平方损失时需要拟合的w_i^*。

【练习】当样本i计算得到的残差$r_i=0$时，本例中的函数在$w=0$处不可导，请问当前轮应当如何处理模型输出?

【练习】除了梯度下降法之外，还可以使用牛顿法来逼近最值点。请叙述基于牛顿法的GBDT回归算法。

$$
\begin{aligned}
w_i^* &= \tilde{w}_i - F_{m-1}(X_i) \\
&= 0 - \frac{\partial L}{\partial w}\frac{\partial w}{\partial \tilde{w}}\Big|_{\tilde{w}=F_{m-1}(X_i)} \\
&= 0 - \frac{\partial L}{\partial w}\Big|_{\tilde{w}=F_{m-1}(X_i)} \\
&= 0 - \frac{\partial L}{\partial w}\Big|_{w=0}
\end{aligned}
$$

上述的结果与先前的梯度下降结果完全一致，事实上这两种观点在本质上没有任何区别，只是损失函数本身进行了平移，下图展示了它们之间的联系。



GBDT的特征重要性

在sklearn实现的GBDT中，特征重要性的计算方式与随机森林相同，即利用相对信息增益来度量单棵树上的各特征特征重要性，再通过对所有树产出的重要性得分进行简单平均来作为最终的特征重要性。

2. 用于分类的GBDT

CART树能够同时处理分类问题和回归问题，但是对于多棵CART进行分类任务的集成时，我们并不能将树的预测结果直接进行类别加和。在GBDT中，我们仍然使用回归树来处理分类问题，那此时拟合的对象和流程又是什么呢？

对于K分类问题，我们假设得到了K个得分$F_{1i},\dots,F_{Ki}$来代表样本i属于对应类别的相对可能性，那么在进行Softmax归一化后，就能够得到该样本属于这些类别的概率大小。其中，属于类别k的概率即为$\frac{e^{F_{ki}}}{\sum_{c=1}^{K}e^{F_{ci}}}$。此时，我们就能够使用多分类的交叉熵函数来计算模型损失，设$\mathbf{y}_i=[y_{1i},\dots,y_{Ki}]$为第$i$个样本的类别独热编码，记$\mathbf{F}_i=[F_{1i},\dots,F_{Ki}]$，则该样本的损失为

$$
L(\mathbf{y}_i,\mathbf{F}_i)=-\sum_{c=1}^{K}y_{ci}\log\frac{e^{F_{ci}}}{\sum_{\tilde{c}=1}^{K}e^{F_{\tilde{c}i}}}
$$

上述的K个得分可以由K棵回归树通过集成学习得到，树的生长目标正是使得上述的损失最小化。记第m轮中K棵树对第i个样本输出的得分为$\mathbf{h}_i^{(m)}=[h_{1i}^{(m)},\dots,h_{Ki}^{(m)}]$，则此时$\mathbf{F}_i^{(m)}=\mathbf{F}_i^{(m-1)}+\mathbf{h}_i^{(m)}$。与GBDT处理回归问题的思路同理，只需要令损失函数$L(\mathbf{y}_i,\mathbf{F}_i)$在$\mathbf{F}_i=\mathbf{F}_i^{(m-1)}$处梯度下降即可：

$$
\mathbf{F}_i^{*(m)}=\mathbf{F}_i^{(m-1)}-\frac{\partial L}{\partial \mathbf{F}_i}\Big|_{\mathbf{F}_i=\mathbf{F}_i^{(m-1)}}
$$

我们需要计算第二项中每一个梯度元素，即

$$
-\frac{\partial L}{\partial \mathbf{F}_i}\Big|_{\mathbf{F}_i=\mathbf{F}_i^{(m-1)}}=[-\frac{\partial L}{\partial F_{1i}}\Big|_{\mathbf{F}_i=\mathbf{F}_i^{(m-1)}},\dots,-\frac{\partial L}{\partial F_{Ki}}\Big|_{\mathbf{F}_i=\mathbf{F}_i^{(m-1)}}]
$$

对于第k个元素有

$$
\begin{aligned}
-\frac{\partial L}{\partial F_{ki}}\Big|_{\mathbf{F}_i=\mathbf{F}_i^{(m-1)}} &= \frac{\partial}{\partial F_{ki}}\sum_{c=1}^{K}y_{ci}\log\frac{e^{F_{ci}}}{\sum_{\tilde{c}=1}^{K}e^{F_{\tilde{c}i}}}\Big|_{\mathbf{F}_i=\mathbf{F}_i^{(m-1)}} \\
&= \frac{\partial}{\partial F_{ki}}\sum_{c=1}^{K}y_{ci}F_{ki}\Big|_{\mathbf{F}_i=\mathbf{F}_i^{(m-1)}} - \frac{\partial}{\partial F_{ki}}\sum_{c=1}^{K}y_{ci}\log[\sum_{\tilde{c}=1}^{K}e^{F_{\tilde{c}i}}]\Big|_{\mathbf{F}_i=\mathbf{F}_i^{(m-1)}} \\
&= y_{ki} - \frac{\partial}{\partial F_{ki}}\sum_{c=1}^{K}y_{ci}\log[\sum_{\tilde{c}=1}^{K}e^{F_{\tilde{c}i}}]\Big|_{\mathbf{F}_i=\mathbf{F}_i^{(m-1)}}
\end{aligned}
$$

由于在上式的第二项里，K个y_{ci}中只有一个为1，且其余为0，从而得到

$$
\begin{aligned}
-\frac{\partial L}{\partial F_{ki}}\Big|_{\mathbf{F}_i=\mathbf{F}_i^{(m-1)}} &= y_{ki} - \frac{\partial}{\partial F_{ki}}\log[\sum_{\tilde{c}=1}^{K}e^{F_{\tilde{c}i}}]\Big|_{\mathbf{F}_i=\mathbf{F}_i^{(m-1)}} \\
&= y_{ki} - \frac{e^{F_{ki}^{(m-1)}}}{\sum_{c=1}^{K}e^{F_{ci}^{(m-1)}}}
\end{aligned}
$$

此时，K棵回归树的学习目标为：

$$
\begin{aligned}
\mathbf{h}_i^{*(m)} &= \mathbf{F}_i^{*(m)} - \mathbf{F}_i^{(m-1)} \\
&= -\frac{\partial L}{\partial \mathbf{F}_i}\Big|_{\mathbf{F}_i=\mathbf{F}_i^{(m-1)}} \\
&= [y_{1i} - \frac{e^{F_{1i}^{(m-1)}}}{\sum_{c=1}^{K} e^{F_{ci}^{(m-1)}}}, \ldots, y_{Ki} - \frac{e^{F_{Ki}^{(m-1)}}}{\sum_{c=1}^{K} e^{F_{ci}^{(m-1)}}}]
\end{aligned}
$$

同时，为了减缓模型的过拟合现象，模型在第m轮实际的$\mathbf{F}_i^{*(m)}$为$\mathbf{F}_i^{(m-1)} + \eta \mathbf{h}_i^{*(m)}$。

由于每一轮都需要进行K棵树的拟合，因此GBDT在处理多分类时的速度较慢。事实上，我们可以利用概率和为1的性质，将K次拟合减少至$K-1$次拟合，这在处理类别数较少的分类问题时，特别是在处理二分类问题时，是非常有用的。

具体来说，此时我们需要$K-1$个得分，记为$F_{1i}, \ldots, F_{(K-1)i}$，则样本相应属于$K$个类别的概率值可表示为

$$
[\frac{e^{F1i}}{1+\sum_{c=1}^{K-1} e^{Fci}}, \ldots, \frac{e^{F(K-1)i}}{1+\sum_{c=1}^{K-1} e^{Fci}}, \frac{1}{1+\sum_{c=1}^{K-1} e^{Fci}}]
$$

当$K \geq 3$时，仍然使用独热编码来写出损失函数：

$$
L(F_{1i}, \ldots, F_{(K-1)i}) = y_{Ki} \log[1 + \sum_{c=1}^{K-1} e^{Fci}] - \sum_{c=1}^{K-1} y_{ci} \log \frac{e^{Fci}}{\sum_{c=1}^{K} e^{Fci}}
$$

类似地记$\mathbf{F}_i = [F_{1i}, \ldots, F_{(K-1)i}]$，我们可以求出负梯度：

【练习】请验证多分类负梯度的结果。

$$
-\frac{\partial L}{\partial F_{ki}}\Big|_{\mathbf{F}_i=\mathbf{F}_i^{(m-1)}} = \begin{cases} -\dfrac{e^{F_{ki}^{(m-1)}}}{\sum_{c=1}^{K-1} e^{F_{ci}^{(m-1)}}} & y_{Ki}=1 \\ y_{ki} - \dfrac{e^{F_{ki}^{(m-1)}}}{\sum_{c=1}^{K-1} e^{F_{ci}^{(m-1)}}} & y_{Ki}=0 \end{cases}
$$

当$K=2$时，不妨规定$y_i \in \{0,1\}$，此时损失函数可简化为

$$
L(F_i) = -y_i \log \frac{e^{F_i}}{1+e^{F_i}} - (1-y_i) \log \frac{1}{1+e^{F_i}}
$$

负梯度为

【练习】请验证二分类负梯度的结果。

$$
-\frac{\partial L}{\partial F_i}\Big|_{F_i=F_i^{(m-1)}} = y_i - \frac{e^{F_i}}{1+e^{F_i}}
$$

最后，我们可以使用各个类别在数据中的占比情况来初始化$\mathbf{F}^{(0)}$。具体地说，设各类别比例为$p_1, \ldots, p_K$（$K \geq 3$），我们希望初始模型的参数$F_1^{(0)}, \ldots, F_{K-1}^{(0)}$满足

$$
[\frac{e^{F_{1i}^{(0)}}}{1+\sum_{c=1}^{K-1} e^{F_{ci}^{(0)}}}, \ldots, \frac{e^{F_{(K-1)i}^{(0)}}}{1+\sum_{c=1}^{K-1} e^{F_{ci}^{(0)}}}, \frac{1}{1+\sum_{c=1}^{K-1} e^{F_{ci}^{(0)}}}] = [p_1, \ldots, p_{K-1}, p_K]
$$

对二分类（0-1分类）而言，设正负样本占比分别为p_1和p_0，则初始模型参数$F^{(0)}$应当满足

【练习】设二分类数据集中正样本比例为10%，请计算模型的初始参数$F^{(0)}$。

$$
[\frac{1}{1+e^{F_i^{(0)}}}, \frac{e^{F_i^{(0)}}}{1+e^{F_i^{(0)}}}] = [p_0, p_1]
$$

> **单调约束（Monotonic Constraints）**
>
> 有时我们会对某个特征或某些特征如何影响模型的输出有先验的知识，例如每天投入在学习的有效时间上越长就越有可能在考试中取得好的成绩，即有效学习时间长度和考试分数是一种单调增的约束关系。许多GBDT的实现（sklearn中的Histogram-Based GBDT、XGBoost和LightGBM）都提供了单调约束的参数选项，有关其在内部的实现原理可以参考本文。

3. XGBoost算法

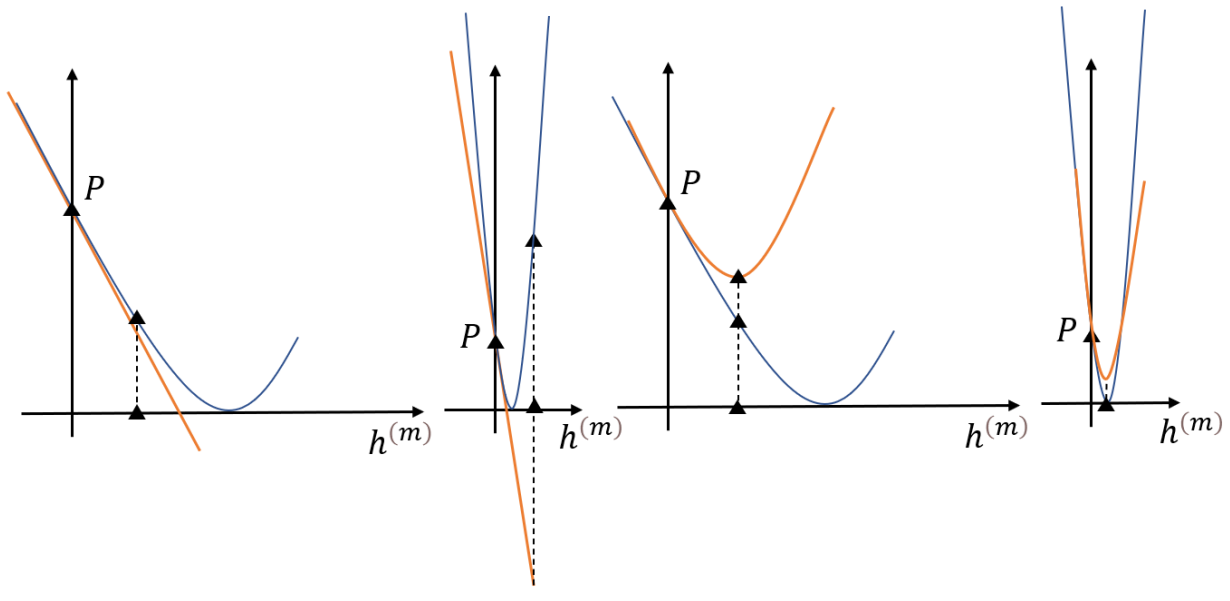
由于树模型较强的拟合能力，我们需要对模型进行正则约束来控制每轮模型学习的进度，除了学习率参数之外，XGBoost还引入了两项作用于损失函数的正则项：首先我们希望树的生长受到抑制而引入γT，其中的T为树的叶子节点个数，γ越大，树就越不容易生长；接着我们希望模型每次的拟合值较小而引入$\frac{1}{2}\lambda \sum_{i=1}^{T} w_i^2$，其中的$w_i$是回归树上第$i$个叶子结点的预测目标值。记第$m$轮中第$i$个样本在上一轮的预测值为$F_i^{(m-1)}$，本轮需要学习的树模型为$h^{(m)}$，此时的损失函数即为

$$
L^{(m)}(h^{(m)}) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j + \sum_{i=1}^{N} L(y_i, F_i^{(m-1)} + h^{(m)}(X_i))
$$

从参数空间的角度而言，损失即为

$$L^{(m)}(F_i^{(m)}) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j + \sum_{i=1}^{N} L(y_i, F_i^{(m)})$$

不同于上一节中GBDT的梯度下降方法，XGBoost直接在$h^{(m)} = 0$处（或$F_i^{(m)} = F_i^{(m-1)}$处）将损失函数近似为一个二次函数，从而直接将该二次函数的顶点坐标作为$h^{*(m)}(X_i)$的值，即具有更小的损失。梯度下降法只依赖损失的一阶导数，当损失的一阶导数变化较大时，使用一步梯度获得的$h^{*(m)}$估计很容易越过最优点，甚至使得损失变大（如子图2所示）；二次函数近似的方法需要同时利用一阶导数和二阶导数的信息，因此对于$h^{*(m)}$的估计在某些情况下会比梯度下降法的估计值更加准确，或说对各类损失函数更有自适应性（如子图3和子图4所示）。



为了得到$h^{*(m)}(X_i)$，记$h_i = h^{(m)}(X_i)$，$\mathbf{h} = [h_1, \ldots, h_N]$，我们需要先将损失函数显式地展开为一个关于$h^{(m)}(X_i)$的二次函数，：

$$\begin{aligned}
L^{(m)}(\mathbf{h}) &= \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j + \sum_{i=1}^{N} L(y_i, F_i^{(m-1)} + h_i) \\
&\approx \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j + \sum_{i=1}^{N} [L(y_i, F_i^{(m-1)}) + \frac{\partial L}{\partial h_i}\Big|_{h_i=0} h_i + \frac{1}{2}\frac{\partial^2 L}{\partial h_i^2}\Big|_{h_i=0} h_i^2] \\
&= \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j + \sum_{i=1}^{N} [\frac{\partial L}{\partial h_i}\Big|_{h_i=0} h_i + \frac{1}{2}\frac{\partial^2 L}{\partial h_i^2}\Big|_{h_i=0} h_i^2] + constant
\end{aligned}$$

由于近似后损失的第二项是按照叶子结点的编号来加和的，而第三项是按照样本编号来加和的，我们为了方便处理，不妨统一将第三项按照叶子结点的编号重排以统一形式。设叶子节点j上的样本编号集合为I_j，记$p_i = \frac{\partial L}{\partial h_i}\Big|_{h_i=0}$且$q_i = \frac{\partial^2 L}{\partial h_i^2}\Big|_{h_i=0}$，忽略常数项后有

【练习】请写出$L^{(m)}(F_i^{(m)})$在$F_i^{(m)} = F_i^{(m-1)}$处的二阶展开。

【练习】试说明不将损失函数展开至更高阶的原因。

【练习】请写出平方损失下的近似损失。

$$\begin{aligned}
\tilde{L}^{(m)}(\mathbf{h}) &= \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j + \sum_{i=1}^{N} [p_i h_i + \frac{1}{2} q_i h_i^2] \\
&= \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j + \sum_{j=1}^{T} [(\sum_{i \in I_j} p_i) w_j + \frac{1}{2}(\sum_{i \in I_j} q_i) w_i^2] \\
&= \gamma T + \sum_{j=1}^{T} [(\sum_{i \in I_j} p_i) w_j + \frac{1}{2}(\sum_{i \in I_j} q_i + \lambda) w_i^2] \\
&= \tilde{L}^{(m)}(\mathbf{w})
\end{aligned}$$

上式的第二个等号是由于同一个叶子节点上的模型输出一定相同，即I_j中样本对应的h_i一定都是w_j。此时，我们将损失统一为了关于叶子节点值$\mathbf{w} = [w_1, \ldots, w_T]$的二次函数，从而可以求得最优的输出值为

$$w_j^* = -\frac{\sum_{i \in I_j} p_i}{\sum_{i \in I_j} q_i + \lambda}$$

当前模型的近似损失（忽略常数项）即为

$$\begin{aligned}
\tilde{L}^{(m)}(\mathbf{w}^*) &= \gamma T + \sum_{j=1}^{T} [-\frac{(\sum_{i \in I_j} p_i)^2}{\sum_{i \in I_j} q_i + \lambda} + \frac{1}{2}\frac{(\sum_{i \in I_j} p_i)^2}{\sum_{i \in I_j} q_i + \lambda}] \\
&= \gamma T - \frac{1}{2}\sum_{j=1}^{T} \frac{(\sum_{i \in I_j} p_i)^2}{\sum_{i \in I_j} q_i + \lambda}
\end{aligned}$$

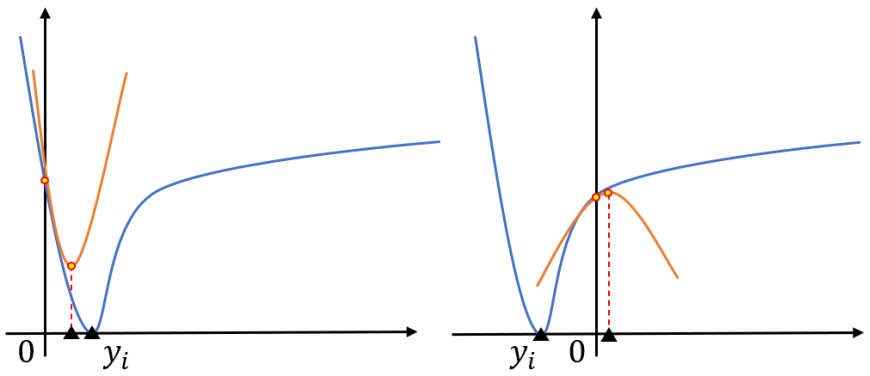
在决策树的一节中，我们曾以信息增益作为节点分裂行为操作的依据，信息增益本质上就是一种损失，增益越大即子节点的平均纯度越高，从而损失就越小。因此我们可以直接将上述的近似损失来作为分裂的依据，即选择使得损失减少得最多的特征及其分割点来进行节点分裂。由于对于某一个节点而言，分裂前后整棵树的损失变化只和该节点I及其左右子节点I_L与L_R的w^*值有关，此时分裂带来的近似损失减少量为

$$
\begin{aligned}
G &= [\gamma T - \frac{1}{2}\frac{(\sum_{i\in I} p_i)^2}{\sum_{i\in I} q_i + \lambda}] - [\gamma(T+1) - \frac{1}{2}\frac{(\sum_{i\in I_L} p_i)^2}{\sum_{i\in I_L} q_i + \lambda} - \frac{1}{2}\frac{(\sum_{i\in I_R} p_i)^2}{\sum_{i\in I_R} q_i + \lambda}] \\
&= \frac{1}{2}[\frac{(\sum_{i\in I_L} p_i)^2}{\sum_{i\in I_L} q_i + \lambda} + \frac{(\sum_{i\in I_R} p_i)^2}{\sum_{i\in I_R} q_i + \lambda} - \frac{(\sum_{i\in I} p_i)^2}{\sum_{i\in I} q_i + \lambda}] - \gamma
\end{aligned}
$$

模型应当选择使得G达到最大的特征和分割点进行分裂。

> **XGBoost的特值处理**
>
> XGBoost不支持分类变量处理，此处的特值是指稀疏值和缺失值，它们的处理方式类似：把0值或缺失值固定，先统一划分至左侧子节点，遍历非0值或非缺失值分割点进行不纯度计算，再统一划分至右侧子节点，又进行非0值或非缺失值分割点的遍历计算，从而得到当前节点当前特征的稀疏值或缺失值默认分配方向以及最佳分割点。特别的是，当训练时特征没有遇到缺失值但预测值出现时，它将会被分配给子节点样本数较多的一侧。

最后我们来重新回到单个样本的损失函数上：由于XGBoost使用的是二阶展开，为了保证函数在拐点处取到的是近似损失的最小值，需要满足二阶导数$q_i > 0$。当损失函数不满足此条件时，h_i^*反而会使得损失上升，即如下图中右侧的情况所示，而使用梯度下降法时并不会产生此问题。因此，我们应当选择在整个定义域上或在y_i临域上二阶导数恒正的损失函数，例如平方损失。



【练习】在下列的三个损失函数$L(y, \hat{y})$中，请选出一个不应作为XGBoost损失的函数并说明理由。

- Root Absolute Error: $\sqrt{|y-\hat{y}|}$
- Squared Log Error: $\frac{1}{2}[\log(\frac{y+1}{\hat{y}+1})]^2$
- Pseudo Huber Error: $\delta^2(\sqrt{1+(\frac{y-\hat{y}}{\delta})^2}-1)$

4. LightGBM算法

LightGBM的GBDT原理与XGBoost的二阶近似方法完全一致，并且在此基础上提出了两个新算法，它们分别是单边梯度采样（GOSS）以及互斥特征绑定（EFB）。

单边梯度采样

在GBDT中，计算出的梯度值绝对值越小则说明样本预测地越是准确，而梯度绝对值越大则说明样本预测的偏离程度越大，因此我们可以考虑对梯度绝对值小的样本进行抽样。具体说，对样本梯度绝对值排序后，先选出Top $a\%$梯度绝对值对应的样本，再从剩下$(1-a)$的样本中抽取$b\%$的样本（此处$b\%$是对于总样本的百分比）。此时，考虑基于均方损失的GBDT回归，记当前节点、左子节点、右子节点的梯度均值为$\bar{g}, \bar{g}_L, \bar{g}_R$，设特征及其分割点为$F, d$，原先的信息增益为

$$
\begin{aligned}
Gain(F,d) &= \frac{1}{N}[\sum_{i=1}^{N}(g_i - \bar{g})^2 - \sum_{i=1}^{N_L}(g_i^{(L)} - \bar{g}_L)^2 - \sum_{i=1}^{N_R}(g_i^{(R)} - \bar{g}_R)^2] \\
&= \frac{1}{N}[(\sum_{i=1}^{N} g_i^2 - N\bar{g}^2) - (\sum_{i=1}^{N_L} {g_i^{(L)}}^2 - N{\bar{g}_L}^2) - (\sum_{i=1}^{N_R} {g_i^{(R)}}^2 - N{\bar{g}_R}^2)] \\
&\propto \frac{1}{N}[\frac{(\sum_{i=1}^{N_L} g_i^{(L)})^2}{N_L} + \frac{(\sum_{i=1}^{N_R} g_i^{(R)})^2}{N_R}]
\end{aligned}
$$

记划分到左子节点对应的a部分样本为A_L、划分到左子节点对应的b部分抽样样本为B_L、划分到右子节点对应的a部分样本为A_R、划分到右子节点对应的b部分抽样样本为B_R。对于抽样部分的梯度和，我们使用$\frac{1-a}{b}$来进行补偿，例如原来从10个样本中划分6个为a部分，从剩下的4个中抽出两个为b部分，那么b部分的样本梯度和估计就是抽出两个样本的梯度和乘以$\frac{1-0.6}{0.2}$。因此，可以写出对应的$\tilde{Gain}(F,d)$为

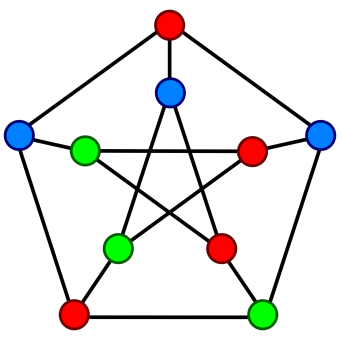
$$
\tilde{Gain}(F,d) = \frac{1}{N}[\frac{(\sum_{i\in A_L} g_i + \frac{1-a}{b}\sum_{i\in B_L} g_i)^2}{N_L} + \frac{(\sum_{i\in A_R} g_i + \frac{1-a}{b}\sum_{i\in B_R} g_i)^2}{N_R}]
$$

互斥特征绑定

实际的数据特征中可能有许多稀疏特征，即其非零值的数量远小于零值的数量，因此希望能够将这些特征进行合并来减少稀疏特征的数量，从而减少直方图构建的时间复杂度。我们将任意两个特征都不同时取非零值的特征集合称为一族互斥特征，数据集中的所有特征可被划分为这样的若干族互斥特征，例如下面就是一族互斥特征。

	特征1	特征2	特征3
样本1	0	1	0
样本2	-1	0	0
样本3	0	0	0

LightGBM提出了将互斥特征合并为单个特征的策略，从而让构建直方图的时间复杂度得以降低，因此需要找到最少的互斥绑定数量，即最少可以划分为几族。遗憾的是这个问题等价于图的着色问题，故它是NP-Hard的，目前并不存在多项式复杂度的解决方案，但我们可以通过近似方法来求解。为什么互斥特征绑定问题与图着色问题等价？如果我们把图的每一个顶点看做特征，将顶点之间是否存在边取决于两个特征是否存在同时为非零值的情况，若是则连接，那么此时没有边的顶点则代表他们之间满足互斥条件，将其涂上同种颜色作为同一族互斥特征，而寻找最少的绑定数量即是要寻找图的最少着色数。下图展示了Petersen图最少需要三种着色数。



【练习】请求出顶点最大度（即最多邻居数量）为d的无向图在最差和最好情况下需要多少种着色数，同时请构造对应的例子。

【练习】在最差情况下LightGBM会生成几族互斥特征？这种情况的发生需要满足什么条件？

在实际操作中，由于严格互斥的特征数量可能还并不算多，但是几乎互斥的特征数量却很多，若存在一个样本使得两个特征同时为非零值则称它们存在一次冲突，所谓几乎互斥即一族特征之间的冲突总数不超过给定的最大冲突数K，此时即使两个顶点之间存在边的连接，只要新加入的顶点能够使得这族特征满足几乎互斥的条件，那么就仍然可进行合并（或着相同颜色），如果此时新顶点与任意一族特征都不满足几乎互斥，那么将自身作为新的一族互斥特征集合的第一个元素（或着新的颜色）。

上述的讨论解决了特征绑定的问题，但我们只是将互斥特征放在了同一个集合里，还没有解决特征合并的问题。直观上说，我们需要用一个特征来表示多个特征时，要求新特征关于原特征族是可辨识的，即存在一一对应的关系。设需要合并的特征为$F_1,\dots,F_m$，它们对应的箱子分割点编号为$B_{i1},\dots,B_{ik_i}(i=1,\dots,m)$。由稀疏性，这里假设$B_{i1}$是0对应的箱子。对于样本$s$而言，如果其对应的特征都为0时，则投放至$\tilde{B}_1$号，若第$i$个特征非0，且其原特征对应的所在箱子为$B_{ij}$，则投放至$\tilde{B}_k$号，其中

$$k=j+\sum_{p=1}^{i-1}k_p$$

对于上述的互斥特征绑定算法而言，我们确实能够对原数据集的特征进行互斥划分，也提取得到了新的直方图分割点，但考虑如下的情况：特征一和特征二是一族互斥特征，当遍历分割点位于特征一对应的非零区域时，此时右侧的点位对应所有的样本被划入右子节点，可此时划入右子节点的特征二非零值，由于互斥特性，本质上其特征一的值还是零，那么这种划分方法与不进行特征绑定单独考虑特征一相同位置的分割点，它们所计算出的信息增益值由于样本划分不同而会产生差异，这与论文中所描述的互斥特征绑定算法能够无损地提高性能不一致。如果有读者清楚其中缘由，欢迎对教程本段内容作出改进或补充说明，在此感谢。

知识回顾

1. GBDT和梯度下降方法有什么联系？
2. 请叙述GBDT用于分类问题的算法流程。
3. XGBoost和GBDT树有何异同？（可从目标损失、近似方法、分裂依据等方面考虑）
4. 请叙述LightGBM中GOSS和EFB的作用及算法流程。
