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：

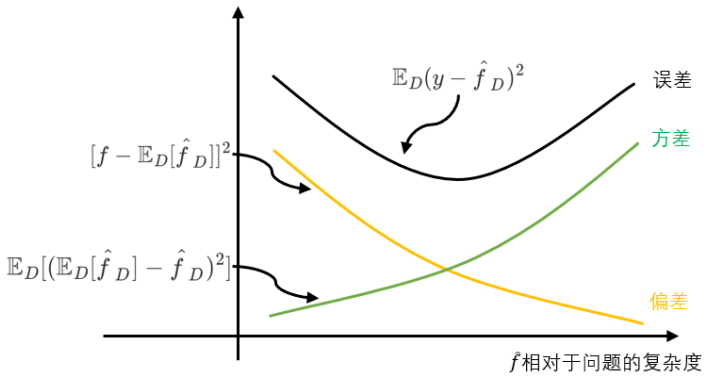
【练习】左式第四个等号为何成立？

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

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。

【练习】有人说Bias-Variance Tradeoff就是指"一个模型要么具有大的偏差，要么具有大的方差"，你认为这种说法对吗？你能否对"偏差-方差权衡"现象做出更准确的表述？



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

【练习】假设总体有100个样本，每轮利用bootstrap抽样从总体中得到10个样本（即可能重复），请求出所有样本都被至少抽出过一次的期望轮数。（通过本文介绍的方法，我们还能得到轮数方差的bound）

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

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

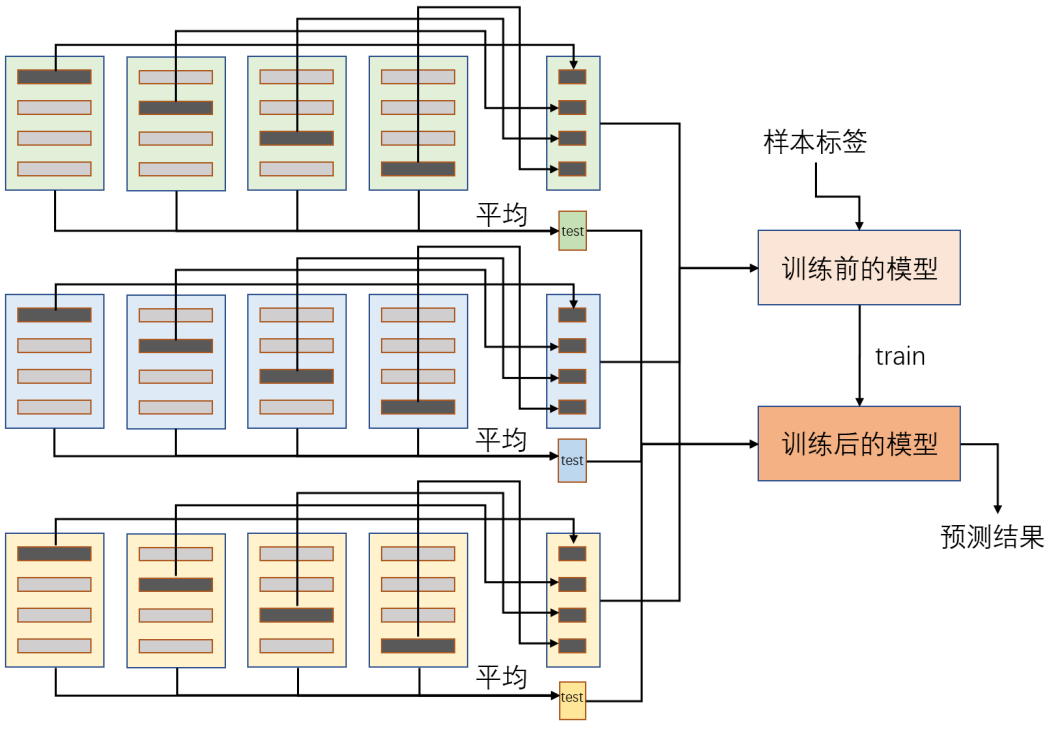
boosting是一种串行集成方法，假设第i个基模型的输出是$\hat{f}^{(i)}(\mathbf{X})$，则总体模型的输出为$\sum_{i=1}^{M} \alpha_i \hat{f}^{(i)}(\tilde{X})$。boosting算法在拟合第$T$个学习器时，已经获得了前$T-1$个学习器的集成输出$\sum_{i=1}^{T-1} \alpha_i \hat{f}^{(i)}(\mathbf{X})$，对于损失函数$L(y,\hat{y})$，当前轮需要优化的目标即为使得$L(y, \alpha_T \hat{f}^{(T)}(\mathbf{X}) + \sum_{i=1}^{T-1} \alpha_i \hat{f}^{(i)}(\mathbf{X}))$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。

【练习】对于stacking和blending集成而言，若m个基模型使用k折交叉验证，此时分别需要进行几次训练和几次预测？



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_。

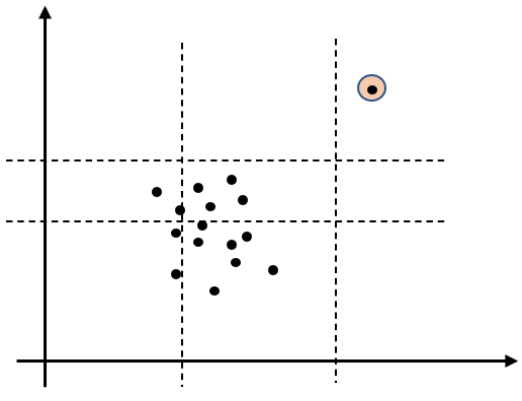
【练习】r2_score和均方误差的区别是什么？它具有什么优势？

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。

【练习】假设使用闵氏距离来度量两个嵌入向量之间的距离，此时对叶子节点的编号顺序会对距离的度量结果有影响吗？

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。



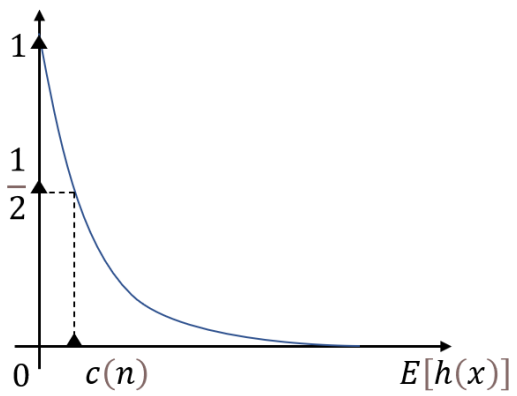
对于n个样本而言，我们可以构建一棵在每个分支进行特征大小判断的树来将样本分派到对应的叶子节点，为了定量刻画异常情况，在这篇文献中证明了树中的平均路径（即树的根节点到叶子结点经过的节点数）长度c为

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

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