机器学习算法系列（34）：使用Sklearn进行集成学习（理论）

一、前言

很多人在竞赛（Kaggle，天池等）或工程实践中使用了集成学习（例如，RF、GTB等），确实也取得了不错的效果，在保证准确度的同时也提升了模型防止过拟合的能力。但是，我们真的用对了集成学习吗?

sklearn提供了sklearn.ensemble库，支持众多集成学习算法和模型。恐怕大多数人使用这些工具时，要么使用默认参数，要么根据模型在测试集上的性能试探性地进行调参（当然，完全不懂的参数还是不动算了），要么将调参的工作丢给调参算法（网格搜索等）。这样并不能真正地称为“会”用sklearn进行集成学习。

我认为，学会调参是进行集成学习工作的前提。然而，第一次遇到这些算法和模型时，肯定会被其丰富的参数所吓到，要知道，教材上教的伪代码可没这么多参数啊！！！没关系，暂时，我们只要记住一句话：参数可分为两种，一种是影响模型在训练集上的准确度或影响防止过拟合能力的参数；另一种不影响这两者的其他参数。模型在样本总体上的准确度（后简称准确度）由其在训练集上的准确度及其防止过拟合的能力所共同决定，所以在调参时，我们主要对第一种参数进行调整，最终达到的效果是：模型在训练集上的准确度和防止过拟合能力的大和谐！

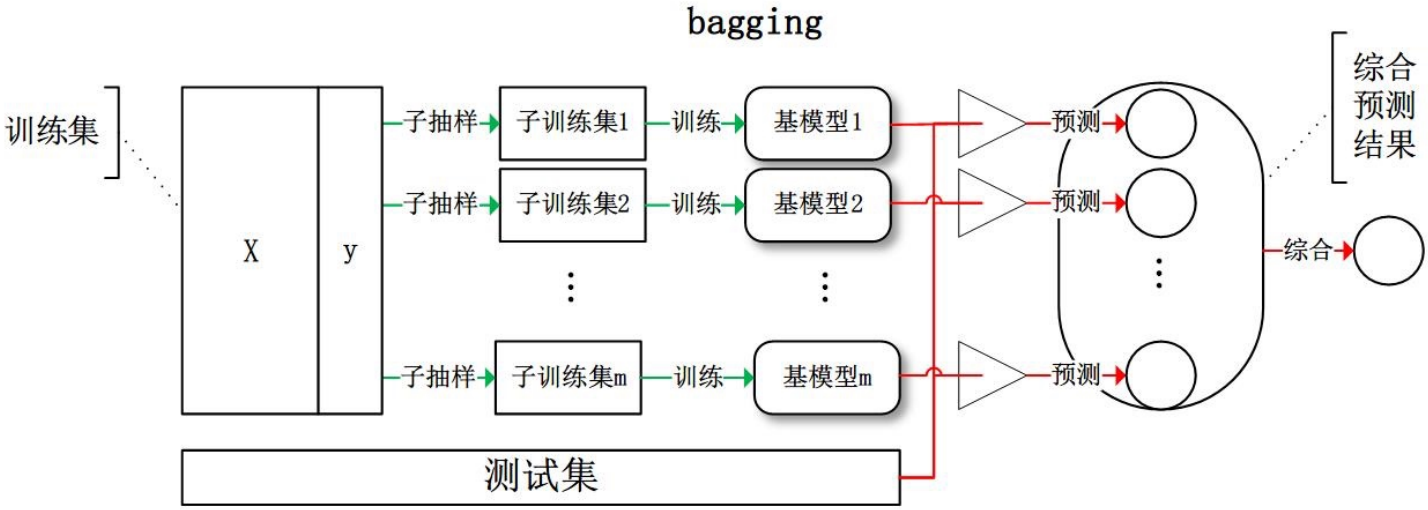
本篇博文将详细阐述模型参数背后的理论知识，在下篇博文中，我们将对最热门的两个模型Random Forrest和Gradient Tree Boosting（含分类和回归，所以共4个模型）进行具体的参数讲解。如果你实在无法静下心来学习理论，你也可以在下篇博文中找到最直接的调参指导，虽然我不赞同这么做。

二、集成学习是什么?

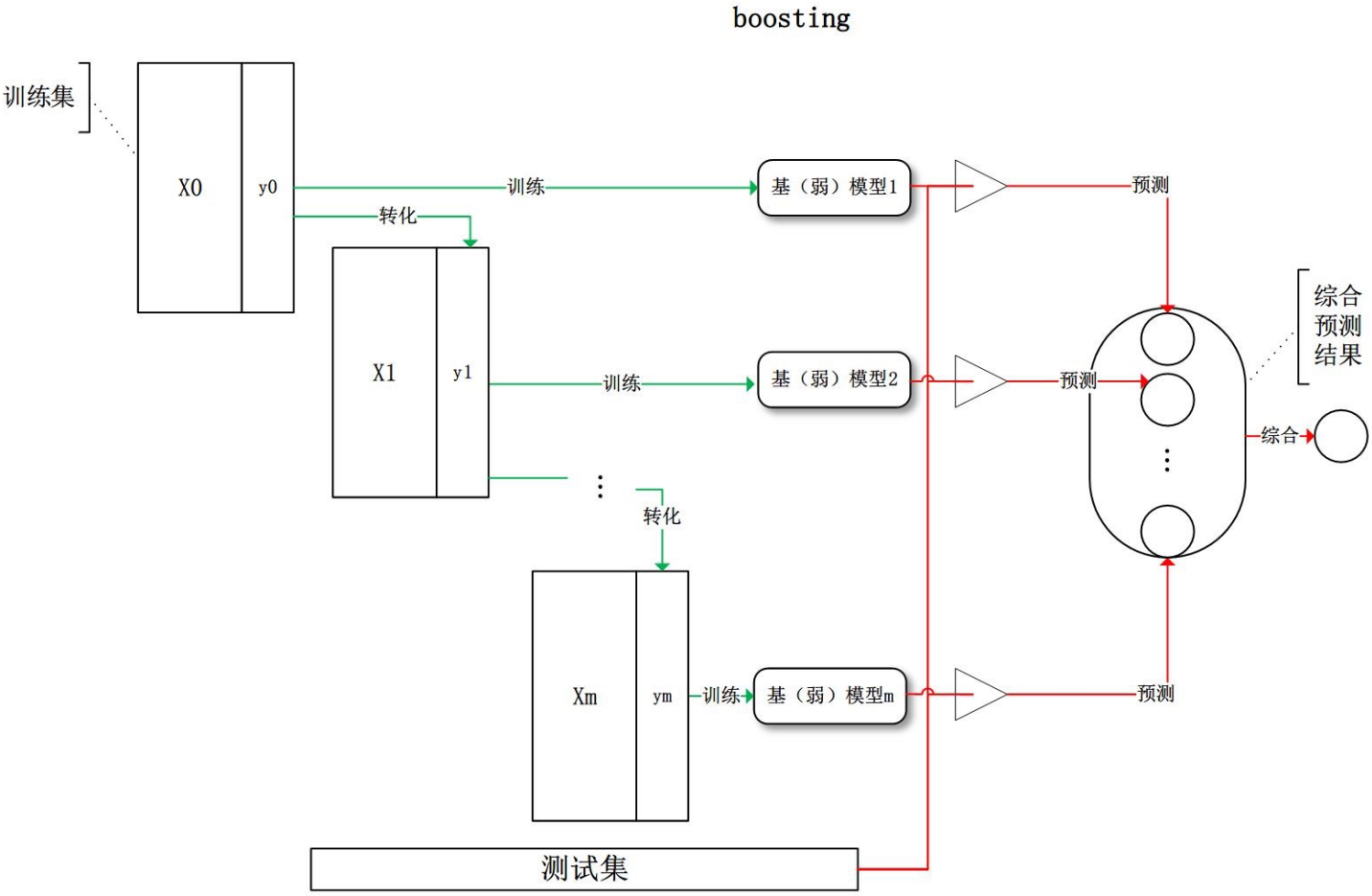
我们还是花一点时间来说明一下集成学习是什么，如果对此有一定基础的同学可以跳过本节。简单来说，集成学习是一种技术框架，其按照不同的思路来组合基础模型，从而达到其利断金的目的。

目前，有三种常见的集成学习框架：bagging，boosting和stacking。国内，南京大学的周志华教授对集成学习有很深入的研究，其在09年发表的一篇概述性论文《Ensemble Learning》对这三种集成学习框架有了明确的定义，概括如下：

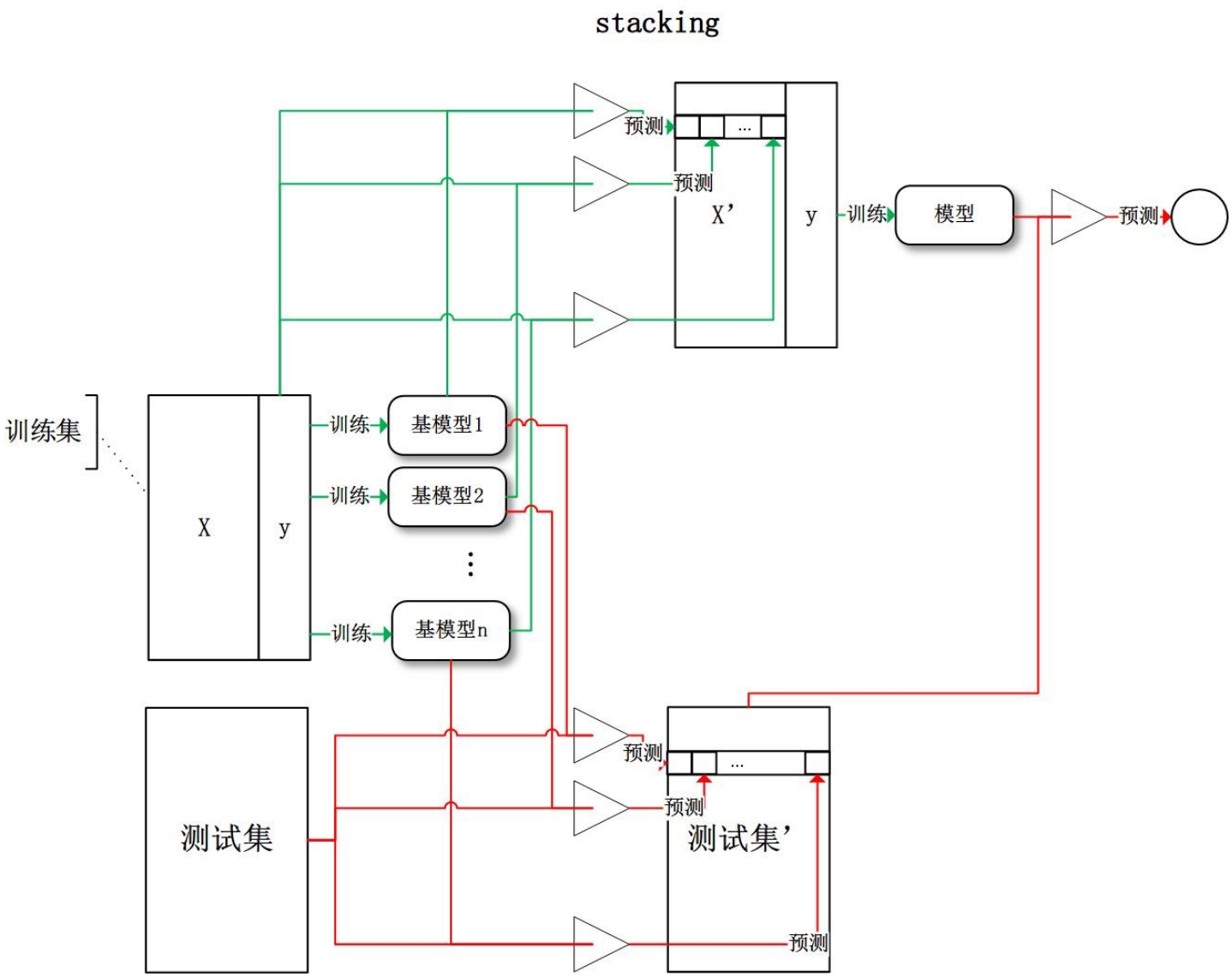
- bagging：从训练集从进行子抽样组成每个基模型所需要的子训练集，对所有基模型预测的结果进行综合产生最终的预测结果：



- boosting：训练过程为阶梯状，基模型按次序一一进行训练（实现上可以做到并行），基模型的训练集按照某种策略每次都进行一定的转化。对所有基模型预测的结果进行线性综合产生最终的预测结果：



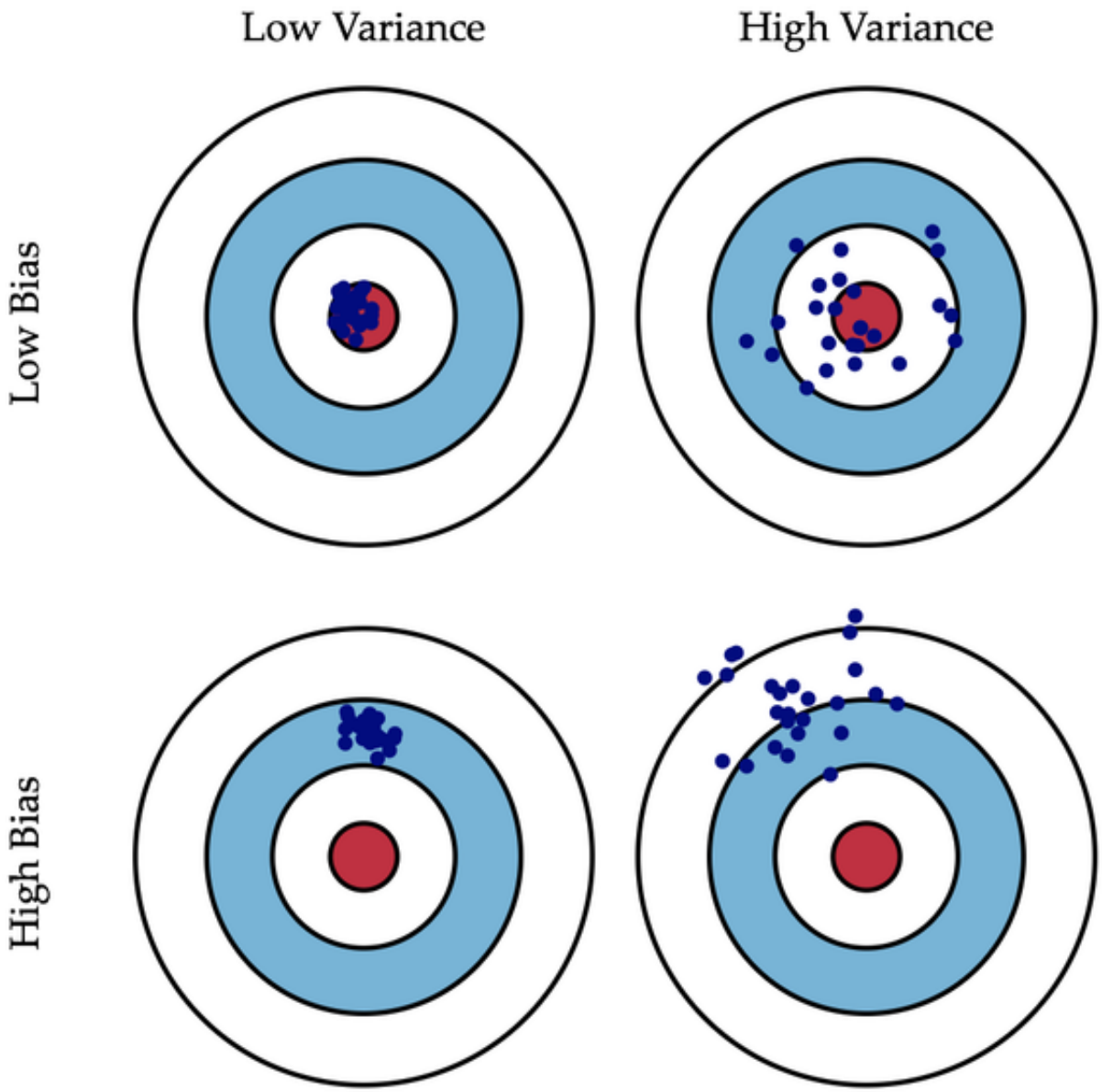
- stacking：将训练好的所有基模型对训练基进行预测，第j个基模型对第i个训练样本的预测值将作为新的训练集中第i个样本的第j个特征值，最后基于新的训练集进行训练。同理，预测的过程也要先经过所有基模型的预测形成新的测试集，最后再对测试集进行预测：



有了这些基本概念之后，直觉将告诉我们，由于不再是单一的模型进行预测，所以模型有了“集思广益”的能力，也就不容易产生过拟合现象。但是，直觉是不可靠的，接下来我们将从模型的偏差和方差入手，彻底搞清楚这一问题。

三、偏差和方差

广义的偏差（bias）描述的是预测值和真实值之间的差异，方差（variance）描述距的是预测值作为随机变量的离散程度。《Understanding the Bias-Variance Tradeoff》当中有一副图形象地向我们展示了偏差和方差的关系：



3.1 模型的偏差和方差

模型的偏差是一个相对来说简单的概念：训练出来的模型在训练集上的准确度。

定义随机变量的值的差异是计算方差的前提条件，通常来说，我们遇到的都是数值型的随机变量，数值之间的差异再明显不过（减法运算）。但是，模型的差异性呢？我们可以理解模型的差异性为模型的结构差异，例如：线性模型中权值向量的差异，树模型中树的结构差异等。在研究模型方差的问题上，我们并不需要对方差进行定量计算，只需要知道其概念即可。

研究模型的方差有什么现实的意义呢？我们认为方差越大的模型越容易过拟合：假设有两个训练集A和B，经过A训练的模型Fa与经过B训练的模型Fb差异很大，这意味着Fa在类A的样本集合上有更好的性能，而Fb反之，这便是我们所说的过拟合现象。

我们常说集成学习框架中的基模型是弱模型，通常来说弱模型是偏差高（在训练集上准确度低）方差小（防止过拟合能力强）的模型。但是，并不是所有集成学习框架中的基模型都是弱模型。bagging和stacking中的基模型为强模型（偏差低方差高），boosting中的基模型为弱模型。

在bagging和boosting框架中，通过计算基模型的期望和方差，我们可以得到模型整体的期望和方差。为了简化模型，我们假设基模型的权重、方差及两两间的相关系数相等。由于bagging和boosting的基模型都是线性组成的，那么有：

$$\mathrm{E}(F) = E\left(\sum_i^m \gamma_i * f_i\right)$$

$$= \sum_i^m \gamma_i * E(f_i)$$

$$= \gamma * \sum_i^m E(f_i)$$

$$\mathrm{Var}(F) = Var\left(\sum_i^m \gamma_i * f_i\right)$$

$$= Cov\left(\sum_i^m \gamma_i * f_i, \sum_i^m \gamma_i * f_i\right)$$

$$= \sum_i^m \gamma_i^2 * Var(f_i) + \sum_i^m \sum_{j \neq i}^m 2 * \rho * \gamma_i * \gamma_j * \sqrt{Var(f_i)} * \sqrt{Var(f_j)}$$

$$= m^2 * \gamma^2 * \sigma^2 * \rho + m * \gamma^2 * \sigma^2 * (1 - \rho)$$

3.2 bagging的偏差和方差

对于bagging来说，每个基模型的权重等于1/m且期望近似相等（子训练集都是从原训练集中进行子抽样），故我们可以进一步化简得到：

$$\mathrm{E}(F) = \gamma * \sum_i^m E(f_i)$$

$$= \frac{1}{m} * m * \mu$$

$$= \mu$$

$$\mathrm{Var}(F) = m^2 * \gamma^2 * \sigma^2 * \rho + m * \gamma^2 * \sigma^2 * (1 - \rho)$$

$$= m^2 * \frac{1}{m^2} * \sigma^2 * \rho + m * \frac{1}{m^2} * \sigma^2 * (1 - \rho)$$

$$= \sigma^2 * \rho + \frac{\sigma^2 * (1 - \rho)}{m}$$

根据上式我们可以看到，整体模型的期望近似于基模型的期望，这也就意味着整体模型的偏差和基模型的偏差近似。同时，整体模型的方差小于等于基模型的方差（当相关性为1时取等号），随着基模型数（m）的增多，整体模型的方差减少，从而防止过拟合的能力增强，模型的准确度得到提高。但是，模型的准确度一定会无限逼近于1吗？并不一定，当基模型数增加到一定程度

时，方差公式第二项的改变对整体方差的作用很小，防止过拟合的能力达到极限，这便是准确度的极限了。另外，在此我们还知道了为什么bagging中的基模型一定要为强模型，否则就会导致整体模型的偏差度低，即准确度低。

Random Forest是典型的基于bagging框架的模型，其在bagging的基础上，进一步降低了模型的方差。Random Fores中基模型是树模型，在树的内部节点分裂过程中，不再是将所有特征，而是随机抽样一部分特征纳入分裂的候选项。这样一来，基模型之间的相关性降低，从而在方差公式中，第一项显著减少，第二项稍微增加，整体方差仍是减少。

3.3 boosting的偏差和方差

对于boosting来说，基模型的训练集抽样是强相关的，那么模型的相关系数近似等于1，故我们也可以针对boosting化简公式为：

$$\mathrm{E}(F) = \gamma * \sum_{i}^{m} E(f_i)$$

$$\begin{aligned}\mathrm{Var}(F) &= m^2 * \gamma^2 * \sigma^2 * \rho + m * \gamma^2 * \sigma^2 * (1-\rho) \\ &= m^2 * \gamma^2 * \sigma^2 * 1 + m * \gamma^2 * \sigma^2 * (1-1) \\ &= m^2 * \gamma^2 * \sigma^2\end{aligned}$$

通过观察整体方差的表达式，我们容易发现，若基模型不是弱模型，其方差相对较大，这将导致整体模型的方差很大，即无法达到防止过拟合的效果。因此，boosting框架中的基模型必须为弱模型。

因为基模型为弱模型，导致了每个基模型的准确度都不是很高（因为其在训练集上的准确度不高）。随着基模型数的增多，整体模型的期望值增加，更接近真实值，因此，整体模型的准确度提高。但是准确度一定会无限逼近于1吗？仍然并不一定，因为训练过程中准确度的提高的主要功臣是整体模型在训练集上的准确度提高，而随着训练的进行，整体模型的方差变大，导致防止过拟合的能力变弱，最终导致了准确度反而有所下降。

基于boosting框架的Gradient Tree Boosting模型中基模型也为树模型，同Random Forrest，我们也可以对特征进行随机抽样来使基模型间的相关性降低，从而达到减少方差的效果。

3.4 模型的独立性

聪明的读者这时肯定要问了，如何衡量基模型的独立性？我们说过，抽样的随机性决定了模型的随机性，如果两个模型的训练集抽样过程不独立，则两个模型则不独立。这时便有一个天大的陷阱在等着我们：bagging中基模型的训练样本都是独立的随机抽样，但是基模型却不独立呢？

我们讨论模型的随机性时，抽样是针对于样本的整体。而bagging中的抽样是针对于训练集（整体的子集），所以并不能称其为对整体的独立随机抽样。那么到底bagging中基模型的相关性体

现在哪呢？在知乎问答《为什么说bagging是减少variance，而boosting是减少bias?》中请教用户“过拟合”后，我总结bagging的抽样为两个过程：

- 样本抽样：整体模型F(X1, X2, ..., Xn)中各输入随机变量（X1, X2, ..., Xn）对样本的抽样
- 子抽样：从整体模型F(X1, X2, ..., Xn)中随机抽取若干输入随机变量成为基模型的输入随机变量

假若在子抽样的过程中，两个基模型抽取的输入随机变量有一定的重合，那么这两个基模型对整体样本的抽样将不再独立，这时基模型之间便具有了相关性。

3.5 小结

还记得调参的目标吗：模型在训练集上的准确度和防止过拟合能力的大和谐！为此，我们目前做了一些什么工作呢?

- 使用模型的偏差和方差来描述其在训练集上的准确度和防止过拟合的能力
- 对于bagging来说，整体模型的偏差和基模型近似，随着训练的进行，整体模型的方差降低
- 对于boosting来说，整体模型的初始偏差较高，方差较低，随着训练的进行，整体模型的偏差降低（虽然也不幸地伴随着方差增高），当训练过度时，因方差增高，整体模型的准确度反而降低
- 整体模型的偏差和方差与基模型的偏差和方差息息相关

这下总算有点开朗了，那些让我们抓狂的参数，现在可以粗略地分为两类了：控制整体训练过程的参数和基模型的参数，这两类参数都在影响着模型在训练集上的准确度以及防止过拟合的能力。

四、Gradient Boosting

对基于Gradient Boosting框架的模型的进行调试时，我们会遇到一个重要的概念：损失函数。在本节中，我们将把损失函数的“今生来世”讲个清楚！

基于boosting框架的整体模型可以用线性组成式来描述，其中$h[i](x)$为基模型与其权值的乘积：

$$\mathrm{F}(x)=\sum_{i}^{m} h_i(x)$$

根据上式，整体模型的训练目标是使预测值F(x)逼近真实值y，也就是说要让每一个基模型的预测值逼近各自要预测的部分真实值。由于要同时考虑所有基模型，导致了整体模型的训练变成了一个非常复杂的问题。所以，研究者们想到了一个贪心的解决手段：每次只训练一个基模型。那么，现在改写整体模型为迭代式：

$$F^i(x)=F^{i-1}(x)+h_i(x)$$

这样一来，每一轮迭代中，只要集中解决一个基模型的训练问题：使$F[i](x)$逼近真实值y。

4.1 拟合残差

使Fi逼近真实值，其实就是使hi逼近真实值和上一轮迭代的预测值Fi-1之差，即残差（y-Fi-1）。最直接的做法是构建基模型来拟合残差，在博文《GBDT（MART） 迭代决策树入门教程 | 简介》中，作者举了一个生动的例子来说明通过基模型拟合残差，最终达到整体模型F(x)逼近真实值。

研究者发现，残差其实是最小均方损失函数的关于预测值的反向梯度：

$$-\frac{\partial\left(\frac{1}{2}*\left(y-F_{i-1}(x)\right)^{2}\right)}{\partial F(x)}=y-F_{i-1}(x)$$

也就是说，若$F[i-1](x)$加上拟合了反向梯度的$h[i](x)$得到$F[i](x)$，该值可能将导致平方差损失函数降低，预测的准确度提高！这显然不是巧合，但是研究者们野心更大，希望能够创造出一种对任意损失函数都可行的训练方法，那么仅仅拟合残差是不恰当的了。

4.2 拟合反向梯度

4.2.1 契机：引入任意损失函数

引入任意损失函数后，我们可以定义整体模型的迭代式如下：

$$F^{i}(x)=F^{i-1}(x)+arg\min_{h\in H}\sum_{j}^{n}L\left(y_j,F^{i-1}(x_j)+h_i(x_j)\right)$$

在这里，损失函数被定义为泛函。

4.2.2 难题一：任意损失函数的最优化

对任意损失函数（且是泛函）的最优化是困难的。我们需要打破思维的枷锁，将整体损失函数L'定义为n元普通函数（n为样本容量），损失函数L定义为2元普通函数（记住！！！这里的损失函数不再是泛函！！！）：

$$L'\left(F^{i-1}(x_1),F^{i-1}(x_2),\dots,F^{i-1}(x_n)\right)=\sum_{j}^{n}L\left(y_j,F^{i-1}(x_j)\right)$$

我们不妨使用梯度最速下降法来解决整体损失函数L'最小化的问题，先求整体损失函数的反向梯度：

$$-\frac{\partial L'\left(F^{i-1}(x_1),F^{i-1}(x_2),\dots,F^{i-1}(x_n)\right)}{\partial F^{i-1}(x_j)}=-\frac{\partial\left(y_j,F^{i-1}(x_j)\right)}{\partial F^{i-1}(x_j)}$$

假设已知样本x的当前预测值为$F[i-1](x)$，下一步将预测值按照反向梯度，依照步长为$r[i]$，进行更新：

$$F^i(x) = F^{i-1}(x) - \gamma_i * \frac{\partial\left(y, F^{i-1}(x)\right)}{\partial F^{i-1}(x)}$$

步长r[i]不是固定值，而是设计为：

$$\gamma_i = \underset{\gamma}{\operatorname{argmin}} \sum_j^n L\left(y_j, F^{i-1}(x) - \gamma * \frac{\partial\left(y_j, F^{i-1}(x_j)\right)}{\partial F^{i-1}(x_j)}\right)$$

4.2.3 难题二：无法对测试样本计算反向梯度

问题又来了，由于测试样本中y是未知的，所以无法求反向梯度。这正是Gradient Boosting框架中的基模型闪亮登场的时刻！在第i轮迭代中，我们创建训练集如下：

$$\left\{\left(x_j, -\frac{\partial\left(y_j, F^{i-1}(x_j)\right)}{\partial F^{i-1}(x_j)}\right)\right\}_j^n$$

也就是说，让基模型拟合反向梯度函数，这样我们就可以做到只输入x这一个参数，就可求出其对应的反向梯度了（当然，通过基模型预测出来的反向梯度并不是准确的，这也提供了泛化整体模型的机会）。

综上，假设第i轮迭代中，根据新训练集训练出来的基模型为$f[i](x)$，那么最终的迭代公式为：

$$F^i(x) = F^{i-1}(x) - \gamma_i * f_i(x)$$

$$\gamma_i = \underset{\gamma}{\operatorname{argmin}} \sum_j^n L(y_j, F^{i-1}(x) - \gamma * f_i(x))$$

4.3 常见的损失函数

ls：最小均方回归中用到的损失函数。在之前我们已经谈到，从拟合残差的角度来说，残差即是该损失函数的反向梯度值（所以又称反向梯度为伪残差）。不同的是，从拟合残差的角度来说，步长是无意义的。该损失函数是sklearn中Gradient Tree Boosting回归模型默认的损失函数。

deviance：逻辑回归中用到的损失函数。熟悉逻辑回归的读者肯定还记得，逻辑回归本质是求极大似然解，其认为样本服从几何分布，样本属于某类别的概率可以logistic函数表达。所以，如果该损失函数可用在多类别的分类问题上，故其是sklearn中Gradient Tree Boosting分类模型默认的损失函数。

exponential：指数损失函数，表达式为：

$$L\left(y_j, F^{i-1}(x_j)\right) = e^{\left(-y_j * F^{i-1}(x_j)\right)}$$

对该损失函数求反向梯度得：

$$-\frac{\partial\left(y_j, F^{i-1}(x_j)\right)}{\partial F^{i-1}(x_j)} = y_j * e^{\left(-y_j * F^{i-1}(x_j)\right)}$$

这时，在第i轮迭代中，新训练集如下：

$$\left\{\left(x_j, y_j * e^{\left(-y_j * F^{i-1}(x_j)\right)}\right)\right\}_j^n$$

脑袋里有什么东西浮出水面了吧？让我们看看Adaboost算法中，第i轮迭代中第j个样本权值的更新公式：

$$w_{i,j} = \frac{e^{\left(-y_j * F^{i-1}(x_j)\right)}}{\sum_1^n e^{\left(-y_j * F^{i-1}(x_j)\right)}}$$

样本的权值什么时候会用到呢？计算第i轮损失函数的时候会用到：

$$L\left(y_j, F^i(x_j)\right) = e^{\left(-y_j * F^i(x_j) * \frac{e^{\left(-y_j * F^{i-1}(x_j)\right)}}{\sum_1^n e^{\left(-y_j * F^{i-1}(x_j)\right)}}\right)}$$

让我们再回过头来，看看使用指数损失函数的Gradient Boosting计算第i轮损失函数：

$$L\left(y_j, F^i(x_j)\right) = e^{\left(-y_j * F^i(x_j) * e^{\left(-y_j * F^{i-1}(x_j)\right)}\right)}$$

天呐，两个公式就差了一个对权值的归一项。这并不是巧合，当损失函数是指数损失时，Gradient Boosting相当于二分类的Adaboost算法。是的，指数损失仅能用于二分类的情况。

4.4 步子太大容易扯着蛋：缩减

缩减也是一个相对显见的概念，也就是说使用Gradient Boosting时，每次学习的步长缩减一点。这有什么好处呢？缩减思想认为每次走一小步，多走几次，更容易逼近真实值。如果步子迈大了，使用最速下降法时，容易迈过最优点。将缩减代入迭代公式：

$$F^i(x) = F^{i-1}(x) - v * \gamma_i * f_i(x), 0 < v \leq 1$$

缩减需要配合基模型数一起使用，当缩减率v降低时，基模型数要配合增大，这样才能提高模型的准确度。

4.5 初始模型

还有一个不那么起眼的问题，初始模型$F[0](x)$是什么呢？如果没有定义初始模型，整体模型的迭代式一刻都无法进行！所以，我们定义初始模型为：

$$F^0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{j}^{n} L(y_j, \gamma)$$

根据上式可知，对于不同的损失函数来说，初始模型也是不一样的。对所有的样本来说，根据初始模型预测出来的值都一样。

4.5 Gradient Tree Boosting

终于到了备受欢迎的Gradient Tree Boosting模型了！但是，可讲的却已经不多了。我们已经知道了该模型的基模型是树模型，并且可以通过对特征的随机抽样进一步减少整体模型的方差。我们可以在维基百科的Gradient Boosting词条中找到其伪代码实现。

4.6 小结

到此，读者应当很清楚Gradient Boosting中的损失函数有什么意义了。要说偏差描述了模型在训练集准确度，则损失函数则是描述该准确度的间接量纲。也就是说，模型采用不同的损失函数，其训练过程会朝着不同的方向进行！

五、总结

磨刀不误砍柴功，我们花了这么多时间来学习必要的理论，我强调一次：必要的理论！集成学习模型的调参工作的核心就是找到合适的参数，能够使整体模型在训练集上的准确度和防止过拟合的能力达到协调，从而达到在样本总体上的最佳准确度。有了本文的理论知识铺垫，在下篇中，我们将对Random Forest和Gradient Tree Boosting中的每个参数进行详细阐述，同时也有一些小试验证明我们的结论。