# **RF框架参数**

　　　　首先我们关注于RF的Bagging框架的参数。这里可以和GBDT对比来学习。在[scikit-learn 梯度提升树(GBDT)调参小结](http://www.cnblogs.com/pinard/p/6143927.html)中我们对GBDT的框架参数做了介绍。GBDT的框架参数比较多，重要的有最大迭代器个数，步长和子采样比例，调参起来比较费力。但是RF则比较简单，这是因为bagging框架里的各个弱学习器之间是没有依赖关系的，这减小的调参的难度。换句话说，达到同样的调参效果，RF调参时间要比GBDT少一些。

　　　　下面我来看看RF重要的Bagging框架的参数，由于RandomForestClassifier和RandomForestRegressor参数绝大部分相同，这里会将它们一起讲，不同点会指出。

　　　　1) **n\_estimators**: 也就是弱学习器的最大迭代次数，或者说最大的弱学习器的个数。一般来说n\_estimators太小，容易欠拟合，n\_estimators太大，又容易过拟合，一般选择一个适中的数值。默认是100。在实际调参的过程中，我们常常将n\_estimators和下面介绍的参数learning\_rate一起考虑。

　　　　2) **oob\_score** :即是否采用袋外样本来评估模型的好坏。默认识False。个人推荐设置为True，因为袋外分数反应了一个模型拟合后的泛化能力。

　　　　3) **criterion:** 即CART树做划分时对特征的评价标准。分类模型和回归模型的损失函数是不一样的。分类RF对应的CART分类树默认是基尼系数gini,另一个可选择的标准是信息增益。回归RF对应的CART回归树默认是均方差mse，另一个可以选择的标准是绝对值差mae。一般来说选择默认的标准就已经很好的。

　　　　从上面可以看出， RF重要的框架参数比较少，主要需要关注的是 n\_estimators，即RF最大的决策树个数。

# **3.  RF决策树参数**

　　　　下面我们再来看RF的决策树参数，它要调参的参数基本和GBDT相同，如下:

　　　　1) RF划分时考虑的最大特征数**max\_features**: 可以使用很多种类型的值，默认是"None",意味着划分时考虑所有的特征数；如果是"log2"意味着划分时最多考虑log2Nlog2N个特征；如果是"sqrt"或者"auto"意味着划分时最多考虑N−−√N个特征。如果是整数，代表考虑的特征绝对数。如果是浮点数，代表考虑特征百分比，即考虑（百分比xN）取整后的特征数。其中N为样本总特征数。一般来说，如果样本特征数不多，比如小于50，我们用默认的"None"就可以了，如果特征数非常多，我们可以灵活使用刚才描述的其他取值来控制划分时考虑的最大特征数，以控制决策树的生成时间。

　　　　2) 决策树最大深度**max\_depth**: 默认可以不输入，如果不输入的话，决策树在建立子树的时候不会限制子树的深度。一般来说，数据少或者特征少的时候可以不管这个值。如果模型样本量多，特征也多的情况下，推荐限制这个最大深度，具体的取值取决于数据的分布。常用的可以取值10-100之间。

　　　　3) 内部节点再划分所需最小样本数**min\_samples\_split**: 这个值限制了子树继续划分的条件，如果某节点的样本数少于min\_samples\_split，则不会继续再尝试选择最优特征来进行划分。 默认是2.如果样本量不大，不需要管这个值。如果样本量数量级非常大，则推荐增大这个值。

　　　　4) 叶子节点最少样本数**min\_samples\_leaf**: 这个值限制了叶子节点最少的样本数，如果某叶子节点数目小于样本数，则会和兄弟节点一起被剪枝。 默认是1,可以输入最少的样本数的整数，或者最少样本数占样本总数的百分比。如果样本量不大，不需要管这个值。如果样本量数量级非常大，则推荐增大这个值。

　　　　5）叶子节点最小的样本权重和**min\_weight\_fraction\_leaf**：这个值限制了叶子节点所有样本权重和的最小值，如果小于这个值，则会和兄弟节点一起被剪枝。 默认是0，就是不考虑权重问题。一般来说，如果我们有较多样本有缺失值，或者分类树样本的分布类别偏差很大，就会引入样本权重，这时我们就要注意这个值了。

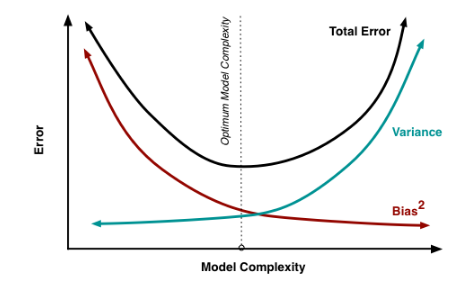
　　　　6) 最大叶子节点数**max\_leaf\_nodes**: 通过限制最大叶子节点数，可以防止过拟合，默认是"None”，即不限制最大的叶子节点数。如果加了限制，算法会建立在最大叶子节点数内最优的决策树。如果特征不多，可以不考虑这个值，但是如果特征分成多的话，可以加以限制，具体的值可以通过交叉验证得到。

　　　　7) 节点划分最小不纯度**min\_impurity\_split:** 这个值限制了决策树的增长，如果某节点的不纯度(基于基尼系数，均方差)小于这个阈值，则该节点不再生成子节点。即为叶子节点 。一般不推荐改动默认值1e-7。

　　　　上面决策树参数中最重要的包括最大特征数max\_features， 最大深度max\_depth， 内部节点再划分所需最小样本数min\_samples\_split和叶子节点最少样本数min\_samples\_leaf。

### 重要参数的意义及设置

  推荐GBDT树的深度：6；（横向比较：DecisionTree/RandomForest需要把树的深度调到15或更高）  
  以下摘自知乎上的一个问答（详见参考文献8），问题和回复都很好的阐述了这个参数设置的数学原理。  
  【问】xgboost/gbdt在调参时为什么树的深度很少就能达到很高的精度？  
  用xgboost/gbdt在在调参的时候把树的最大深度调成6就有很高的精度了。但是用DecisionTree/RandomForest的时候需要把树的深度调到15或更高。用RandomForest所需要的树的深度和DecisionTree一样我能理解，因为它是用bagging的方法把DecisionTree组合在一起，相当于做了多次DecisionTree一样。但是xgboost/gbdt仅仅用梯度上升法就能用6个节点的深度达到很高的预测精度，使我惊讶到怀疑它是黑科技了。请问下xgboost/gbdt是怎么做到的？它的节点和一般的DecisionTree不同吗？  
  【答】  
  这是一个非常好的问题，题主对各算法的学习非常细致透彻，问的问题也关系到这两个算法的本质。这个问题其实并不是一个很简单的问题，我尝试用我浅薄的机器学习知识对这个问题进行回答。  
  一句话的解释，来自周志华老师的机器学习教科书（ 机器学习-周志华）：Boosting主要关注降低偏差，因此Boosting能基于泛化性能相当弱的学习器构建出很强的集成；Bagging主要关注降低方差，因此它在不剪枝的决策树、神经网络等学习器上效用更为明显。  
  随机森林(random forest)和GBDT都是属于集成学习（ensemble learning)的范畴。集成学习下有两个重要的策略Bagging和Boosting。  
  Bagging算法是这样做的：每个分类器都随机从原样本中做有放回的采样，然后分别在这些采样后的样本上训练分类器，然后再把这些分类器组合起来。简单的多数投票一般就可以。其代表算法是随机森林。Boosting的意思是这样，他通过迭代地训练一系列的分类器，每个分类器采用的样本分布都和上一轮的学习结果有关。其代表算法是AdaBoost, GBDT。  
  其实就机器学习算法来说，其泛化误差可以分解为两部分，偏差（bias)和方差(variance)。这个可由下图的式子导出（这里用到了概率论公式D(X)=E(X^2)-[E(X)]^2）。偏差指的是算法的期望预测与真实预测之间的偏差程度，反应了模型本身的拟合能力；方差度量了同等大小的训练集的变动导致学习性能的变化，刻画了数据扰动所导致的影响。这个有点儿绕，不过你一定知道过拟合。  
  如下图所示，当模型越复杂时，拟合的程度就越高，模型的训练偏差就越小。但此时如果换一组数据可能模型的变化就会很大，即模型的方差很大。所以模型过于复杂的时候会导致过拟合。  
  当模型越简单时，即使我们再换一组数据，最后得出的学习器和之前的学习器的差别就不那么大，模型的方差很小。还是因为模型简单，所以偏差会很大。



模型复杂度与偏差方差的关系图

  也就是说，当我们训练一个模型时，偏差和方差都得照顾到，漏掉一个都不行。  
  对于Bagging算法来说，由于我们会并行地训练很多不同的分类器的目的就是降低这个方差(variance) ,因为采用了相互独立的基分类器多了以后，h的值自然就会靠近.所以对于每个基分类器来说，目标就是如何降低这个偏差（bias),所以我们会采用深度很深甚至不剪枝的决策树。  
  对于Boosting来说，每一步我们都会在上一轮的基础上更加拟合原数据，所以可以保证偏差（bias）,所以对于每个基分类器来说，问题就在于如何选择variance更小的分类器，即更简单的分类器，所以我们选择了深度很浅的决策树。