1. 算法原理

决策树-随机森林

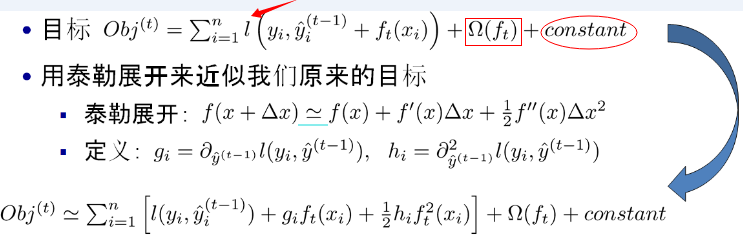
决策树-Adaboost算法-GBDT（CART）--XGboost

XGboost的算法原理和GBDT的思路非常相似，之前提到GBDT每一次的计算都是为了减少上一次的残差，进而在残差减少（负梯度）方向上建立一个新的模型。

而XGboost本质上也是一个GBDT，但是力争将速度和效率发挥到极致，所以叫Extreme GBoosted。

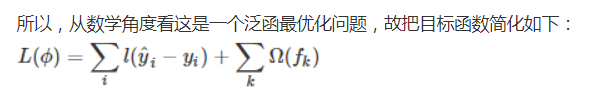
而XGboost与GBDT的最大不同之处就在于目标函数的定义。

每一次迭代，都在现有树的基础上，增加一棵树去拟合前面树的预测结果与真实值之间的残差



1. 损失函数

Logistic损失函数，对数损失函数，平方误差损失函数



损失函数+正则化

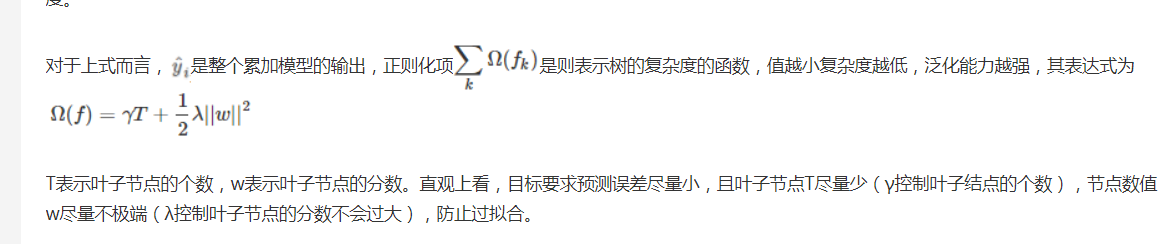
1. 分裂节点算法

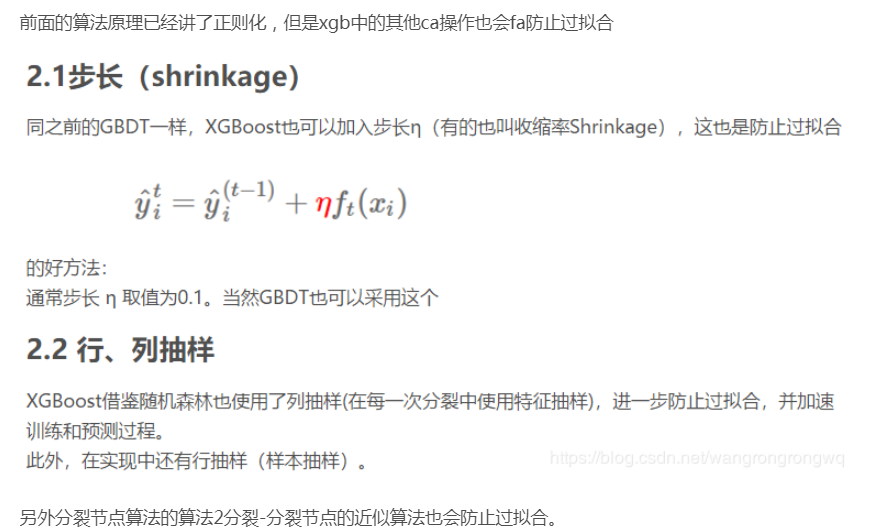
XGBoost每一步选能使分裂后增益最大的分裂点进行分裂。而分裂点的选取之前是枚举所有分割点，这称为精确的贪心法（Exact Greedy Algorithm）.当数据量十分庞大，以致于不能全部放入内存时，Exact Greedy 算法就会很慢。因此XGBoost引入了近似的算法。即对每一个特征进行「值」采样。原来需要对每一个特征的每一个可能分割点进行尝试，采样之后只针对采样的点（分位数）进行分割尝试，这种方法很明显可以减少计算量，采样密度越小计算的越快，拟合程度也会越差，所以采样还可以防止过拟合。

把样本从根分配到叶子结点，就是个排列组合。不同的组合对应的cost不同。求最好的组合你就要try，一味穷举是不可能的，所以才出来贪婪法。不看从头到尾 就看当下节点怎么分配最好。这才有了那个greddy exact 方法，后来还想加速才有了histogram的做法。

原文链接：https://blog.csdn.net/wangrongrongwq/article/details/86755915

1. 正则化





1. 对缺失值的处理

有很多种原因可能导致特征的稀疏（缺失），所以当遇到样本某个维度的特征缺失的时候，就不能知道这个样本会落在左孩子还是右孩子。xgboost处理缺失值的方法和其他树模型不同。根据作者TianqiChen在论文[1]中章节3.4的介绍，xgboost把缺失值当做稀疏矩阵来对待，本身在节点分裂时不考虑缺失值的数值，但确定分裂的特征后，缺失值数据会被分到左子树和右子树呢？本文的处理策略是落在哪个孩子得分高，就放到哪里。如果训练中没有数据缺失，预测时出现了数据缺失，那么默认被分类到右子树。

1. 优缺点

1.损失函数是用泰勒展式二项逼近，而不是像gbdt里的就是一阶导数

2.对树的结构进行了正则化约束，防止模型过度复杂，降低了过拟合的可能性

3.节点分裂的方式不同，gbdt是用的gini系数，xgboost是经过优化推导后的

xgBoosting在传统Boosting的基础上，利用cpu的多线程，引入正则化项，加入剪纸，控制了模型的复杂度。

与GBDT相比，xgBoosting有以下进步：

1）GBDT以传统CART作为基分类器，而xgBoosting支持线性分类器，相当于引入L1和L2正则化项的逻辑回归（分类问题）和线性回归（回归问题）；

2）GBDT在优化时只用到一阶导数，xgBoosting对代价函数做了二阶Talor展开，引入了一阶导数和二阶导数；

3）当样本存在缺失值是，xgBoosting能自动学习分裂方向；

4）xgBoosting借鉴RF的做法，支持列抽样，这样不仅能防止过拟合，还能降低计算；

5）xgBoosting的代价函数引入正则化项，控制了模型的复杂度，正则化项包含全部叶子节点的个数，每个叶子节点输出的score的L2模的平方和。从贝叶斯方差角度考虑，正则项降低了模型的方差，防止模型过拟合；

6）xgBoosting在每次迭代之后，为叶子结点分配学习速率，降低每棵树的权重，减少每棵树的影响，为后面提供更好的学习空间；

7）xgBoosting工具支持并行,但并不是tree粒度上的，而是特征粒度，决策树最耗时的步骤是对特征的值排序，xgBoosting在迭代之前，先进行预排序，存为block结构，每次迭代，重复使用该结构，降低了模型的计算；block结构也为模型提供了并行可能，在进行结点的分裂时，计算每个特征的增益，选增益最大的特征进行下一步分裂，那么各个特征的增益可以开多线程进行；

8）可并行的近似直方图算法，树结点在进行分裂时，需要计算每个节点的增益，若数据量较大，对所有节点的特征进行排序，遍历的得到最优分割点，这种贪心法异常耗时，这时引进近似直方图算法，用于生成高效的分割点，即用分裂后的某种值减去分裂前的某种值，获得增益，为了限制树的增长，引入阈值，当增益大于阈值时，进行分裂；

然而，与LightGBM相比，又表现出了明显的不足：

1）xgBoosting采用预排序，在迭代之前，对结点的特征做预排序，遍历选择最优分割点，数据量大时，贪心法耗时，LightGBM方法采用histogram算法，占用的内存低，数据分割的复杂度更低；

2）xgBoosting采用level-wise生成决策树，同时分裂同一层的叶子，从而进行多线程优化，不容易过拟合，但很多叶子节点的分裂增益较低，没必要进行跟进一步的分裂，这就带来了不必要的开销；LightGBM采用深度优化，leaf-wise生长策略，每次从当前叶子中选择增益最大的结点进行分裂，循环迭代，但会生长出更深的决策树，产生过拟合，因此引入了一个阈值进行限制，防止过拟合.

原文链接：https://blog.csdn.net/mingxiaod/article/details/86063153

<https://blog.csdn.net/mingxiaod/article/details/86063153#_73>

https://www.julyedu.com/question/big/kp\_id/23/ques\_id/987

1. 应用场景

评分系统，智能垃圾邮件识别，广告推荐系统

https://www.cnblogs.com/burton/p/10473199.html

1. Sklearn参数

https://blog.csdn.net/wangrongrongwq/article/details/86755915