20. LightGBM常见问题

1．LightGBM是什么

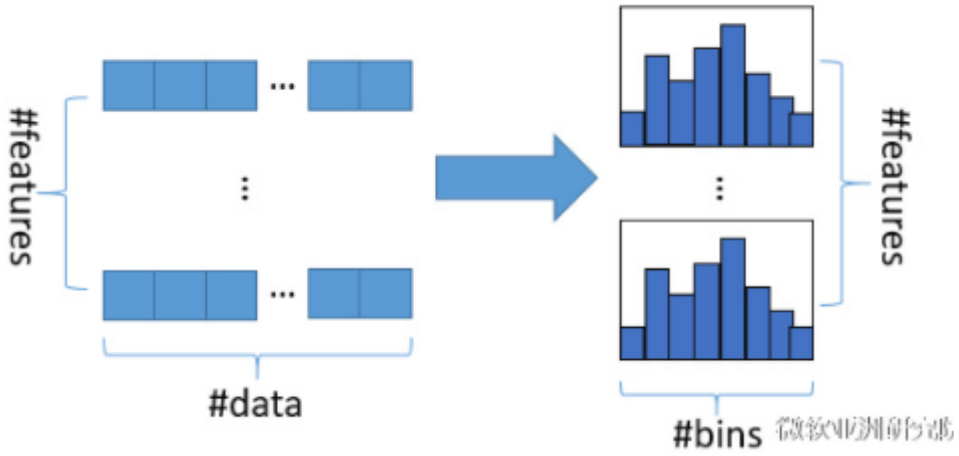
LightGBM （Light Gradient Boosting Machine）(请点击 https://github.com/Microsoft/LightGBM) <https://github.com/Microsoft/LightGBM> 是一个实现GBDT算法的框架，支持高效率的并行训练。LightGBM在Higgs数据集上LightGBM比XGBoost快将近10倍，内存占用率大约为XGBoost的1/6，并且准确率也有提升。GBDT在每一次迭代的时候，都需要遍历整个训练数据多次。如果把整个训练数据装进内存则会限制训练数据的大小；如果不装进内存，反复地读写训练数据又会消耗非常大的时间。尤其面对工业级海量的数据，普通的GBDT算法是不能满足其需求的。LightGBM提出的主要原因就是为了解决GBDT在海量数据遇到的问题，让GBDT可以更好更快地用于工业实践。

2．LightGBM在哪些地方进行了优化 (区别XGBoost)

- **基于Histogram的决策树算法**
- **带深度限制的Leaf-wise的叶子生长策略**
- **直方图做差加速直接**
- **支持类别特征(Categorical Feature)**
- Cache命中率优化
- 基于直方图的稀疏特征优化多线程优化

3．Histogram算法

直方图算法的基本思想是先把连续的浮点特征值离散化成k个整数（其实又是分桶的思想，而这些桶称为bin，比如[0,0.1)→0, [0.1,0.3)→1），同时构造一个宽度为k的直方图。在遍历数据的时候，根据离散化后的值作为索引在直方图中累积统计量，当遍历一次数据后，直方图累积了需要的统计量，然后根据直方图的离散值，遍历寻找最优的分割点。



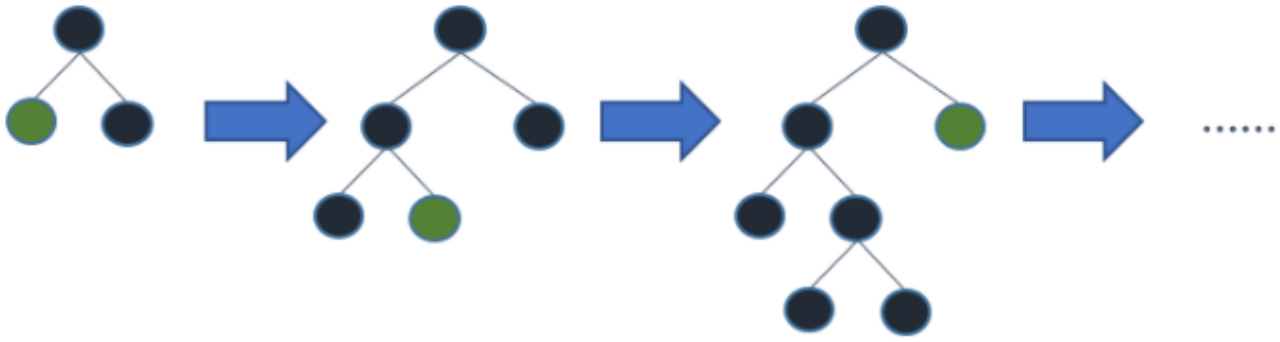
使用直方图算法有很多优点。首先，最明显就是内存消耗的降低，直方图算法不仅不需要额外存储预排序的结果，而且可以只保存特征离散化后的值，而这个值一般用8位整型存储就足够了，内存消耗可以降低为原来的1/8。然后在计算上的代价也大幅降低，预排序算法每遍历一个特征值就需要计算一次分裂的增益，而直方图算法只需要计算k次（k可以认为是常数），时间复杂度从O(#data*#feature)优化到O(k*#features)。

4．带深度限制的Leaf-wise的叶子生长策略

在XGBoost中，树是按层生长的，称为Level-wise tree growth，同一层的所有节点都做分裂，最后剪枝，如下图所示：

Level-wise过一次数据可以同时分裂同一层的叶子，容易进行多线程优化，也好控制模型复杂度，不容易过拟合。但实际上Level-wise是一种低效的算法，因为它不加区分的对待同一层的叶子，带来了很多没必要的开销，因为实际上很多叶子的分裂增益较低，没必要进行搜索和分裂。

在Histogram算法之上，LightGBM进行进一步的优化。首先它抛弃了大多数GBDT工具使用的按层生长 (level-wise)的决策树生长策略，而使用了带有深度限制的按叶子生长 (leaf-wise)算法。



Leaf-wise tree growth

Leaf-wise则是一种更为高效的策略，**每次从当前所有叶子中，找到分裂增益最大的一个叶子，然后分裂，如此循环。**因此同Level-wise相比，在分裂次数相同的情况下，Leaf-wise可以降低更多的误差，得到更好的精度。Leaf-wise的缺点是可能会长出比较深的决策树，产生过拟合。因此LightGBM在Leaf-wise之上增加了一个最大深度的限制，在保证高效率的同时防止过拟合。

5．直方图差加速

LightGBM另一个优化是Histogram（直方图）做差加速。一个容易观察到的现象：一个叶子的直方图可以由它的父亲节点的直方图与它兄弟的直方图做差得到。通常构造直方图，需要遍历该叶子上的所有数据，但直方图做差仅需遍历直方图的k个桶。利用这个方法，LightGBM可以在构造一个叶子的直方图后，可以用非常微小的代价得到它兄弟叶子的直方图，在速度上可以提升一倍。

6．直接支持类别特征

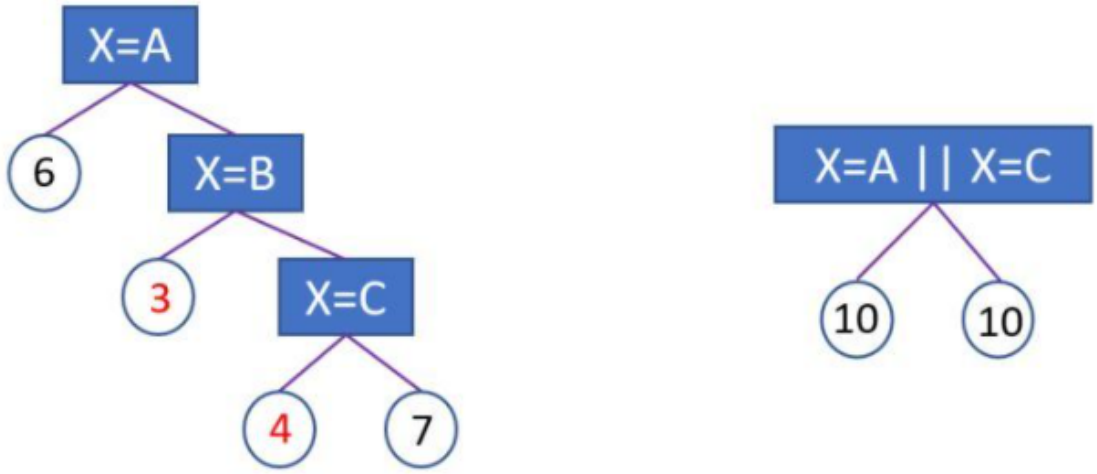
one-hot编码弊端：

one-hot编码是处理类别特征的一个通用方法，然而在树模型中，这可能并不一定是一个好的方法，尤其当类别特征中类别个数很多的情况下。主要的问题是：

①可能无法在这个类别特征上进行切分（即浪费了这个特征）。使用one-hot编码的话，意味着在每一个决策节点上只能使用one vs rest（例如是不是狗，是不是猫等）的切分方式。当类别值很多时，每个类别上的数据可能会比较少，这时候切分会产生不平衡，这意味着切分增益也会很小（比较直观的理解是，不平衡的切分和不切分没有区别）。

②会影响决策树的学习。因为就算可以在这个类别特征进行切分，也会把数据切分到很多零碎的小空间上，如图1左边所示。而决策树学习时利用的是统计信息，在这些数据量小的空间上，统计信息不准确，学习会变差。但如果使用如图1右边的分裂方式，数据会被切分到两个比较大的空间，进一步的学习也会更好。

图1右边叶子节点的含义是X=A或者X=C放到左孩子，其余放到右孩子。

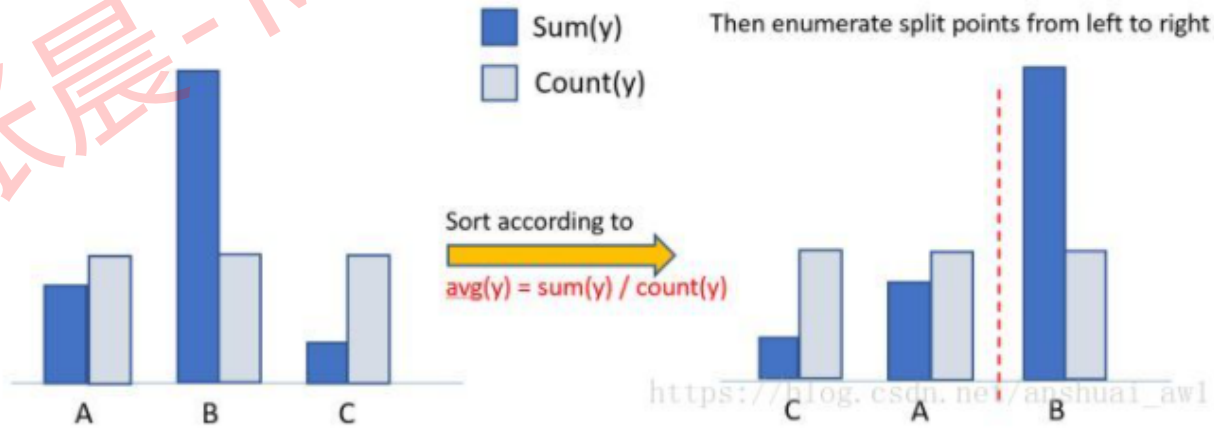


Note: Numbers in circles represent to the #data in that node

LGBM处理分类特征：

为了解决one-hot编码处理类别特征的不足。LGBM采用了Many vs many的切分方式，实现了类别特征的最优切分。用Lightgbm可以直接输入类别特征，并产生如图1右边的效果。在1个k维的类别特征中寻找最优切分，朴素的枚举算法的复杂度是$O(2^k)$，而LGBM采用了如参考文献【1】的方法实现了$O(klogk)$的算法。

算法流程如图2所示：在枚举分割点之前，先把直方图按每个类别的均值进行排序；然后按照均值的结果依次枚举最优分割点。从图2可以看到，Sum(y)/Count(y)为类别的均值。当然，这个方法很容易过拟合，所以在LGBM中加入了很多对这个方法的约束和正则化。



7．LightGBM优点

- 更快的训练速度
- 更低的内存消耗
- 更好的准确率
- 分布式支持，可以快速处理海量数据

https://blog.csdn.net/weixin_41510260/article/details/95378749

<https://blog.csdn.net/weixin_41510260/article/details/95378749>