6. LGB（一）原理

1.LightGBM简介

GBDT (Gradient Boosting Decision Tree) 是机器学习中一个长盛不衰的模型，其主要思想是利用弱分类器（决策树）迭代训练以得到最优模型，该模型具有训练效果好、不易过拟合等优点。

而LightGBM（Light Gradient Boosting Machine）是一个实现GBDT算法的框架，支持高效率的并行训练，并且具有更快的训练速度、更低的内存消耗、更好的准确率、支持分布式可以快速处理海量数据等优点。

1.1 LightGBM提出的动机

常用的机器学习算法，例如神经网络等算法，都可以以mini-batch的方式训练，训练数据的大小不会受到内存限制。

而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。

1.2 XGBoost的缺点

XGBoost缺点：每一次都需要遍历所有的特征，然后计算每一个特征的信息增益(涉及对每一个样本的计算)，然后选择最好的一个特征进行分裂 ，一次分裂计算需要计算N_samples * N_features。

这样的预排序算法的优点是能精确地找到分割点。但是缺点也很明显：

1.2.1空间消耗大

这样的算法需要保存数据的特征值，还保存了特征排序的结果（例如，为了后续快速的计算分割点，保存了排序后的索引），这就需要消耗训练数据两倍的内存。

1.2.2时间上也有较大的开销

在遍历每一个分割点的时候，都需要进行分裂增益的计算，消耗的代价大。

1.3 LightGBM的优化

为了避免上述XGBoost的缺陷，并且能够在不损害准确率的条件下加快GBDT模型的训练速度，lightGBM在传统的GBDT算法上进行了如下优化：

- LightGBM采用直方图算法将遍历样本转变为遍历直方图，极大的降低了时间复杂度
- LightGBM 在训练过程中采用**单边梯度算法**过滤掉梯度小的样本，减少了大量的计算；
- LightGBM 采用了**互斥特征捆绑算法**减少特征的维数。
- LightGBM 采用优化后的**特征并行、数据并行**方法加速计算，当数据量非常大的时候还可以采用投票并行的策略；

2.LightGBM的基本原理

2.1 基于Histogram(直方图)的决策树算法

2.1.1基本思想

一个特征 对应 一个直方图。

将连续值，转为为多个离散区间存储。（将特征的值做一次映射）

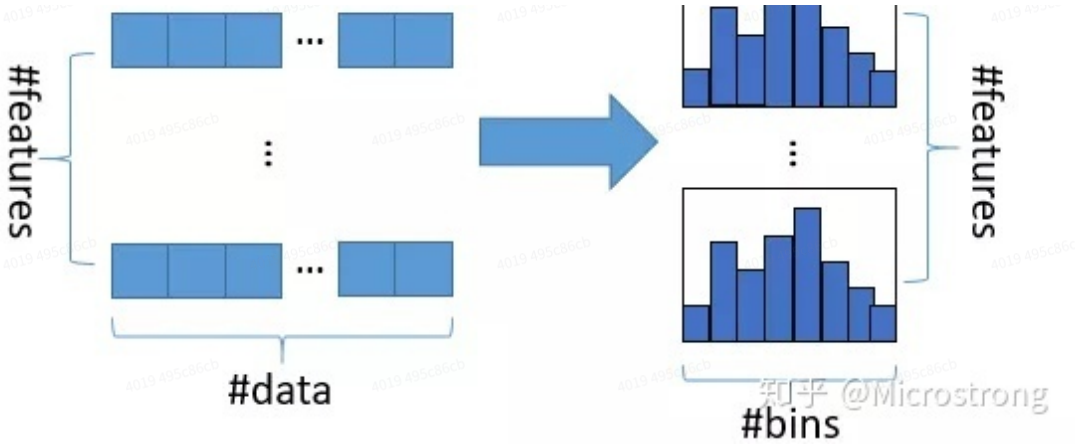
对于一个特征，先把连续的浮点特征值离散化成 k个整数，构造一个宽度为k 的直方图。

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，直方图累积了原始样本的连续值在离散区间出现次数。

每一个直方图得到后，计算信息增益就直接对直方图的区间进行计算。这样不用遍历所有的样本，只需要遍历所有离散的区间值。而离散的区间值数目 远远小于 样本数量。这样就能大大减小时间复杂度。

直接将时间复杂度从O(#data * # feature) 降低到 O(k * #feature),而#data >> k。

然后根据直方图的离散值，遍历寻找最优的分割点。



2.1.2 优点

1.内存占用更小

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

2.计算代价更小

预排序算法XGBoost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LightGBM只需要计算 k次，直接将时间复杂度从O(#data * # feature) 降低到 O(k * #feature),而#data >> k。

2.3 单边梯度采样算法(减少样本数量)

Gradient-based One-Side Sampling 应该被翻译为单边梯度采样（GOSS）。**GOSS算法从减少样本的角度出发**，排除大部分小梯度的样本，仅用剩下的样本计算信息增益，**它是一种在减少数据量和保证精度上平衡的算法。**

2.3.1动机

AdaBoost中，样本权重是数据重要性的指标。然而在GBDT中没有原始样本权重，不能应用权重采样。幸运的是，我们观察到GBDT中每个数据都有不同的梯度值，对采样十分有用。

梯度小的样本，训练误差也比较小，说明数据已经被模型学习得很好了，直接想法就是丢掉这部分梯度小的数据。然而这样做会改变数据的分布，将会影响训练模型的精确度，为了避免此问题，提出了GOSS算法。

2.3.2基本原理

GOSS是一个样本的采样算法，目的是丢弃一些对计算信息增益没有帮助的样本留下有帮助的。根据计算信息增益的定义，梯度大的样本对信息增益有更大的影响。因此，GOSS在进行数据采样的

时候只保留了梯度较大的数据，但是如果直接将所有梯度较小的数据都丢弃掉势必会影响数据的总体分布。

所以，GOSS首先将要进行分裂的特征的所有取值按照绝对值大小降序排序（XGBoost一样也进行了排序，但是LightGBM不用保存排序后的结果)。

选取绝对值最大的 a 个数据。然后在剩下的较小梯度数据中随机选择b个数据。接着讲这b个数据乘常数（1-a)/b, 这样算法就会更关注训练不足的样本，而不会过多的改变原始数据集的分布。最后用这个（a + b）个数据来计算信息增益。

一方面算法将更多的注意力放在训练不足的样本上，另一方面通过乘上权重来防止采样对原始数据分布造成太大的影响。

2.4互斥特征捆绑算法（减小特征维度）

2.4.1动机

高维度的数据往往是稀疏的，这种稀疏性启发我们设计一种无损的方法来减少特征的维度。通常被捆绑的特征都是互斥的（即特征不会同时为非零值，像one-hot），这样两个特征捆绑起来才不会丢失信息。

如果两个特征并不是完全互斥（部分情况下两个特征都是非零值），可以用一个指标对特征不互斥程度进行衡量，称之为冲突比率，当这个值较小时，我们可以选择把不完全互斥的两个特征捆绑，而不影响最后的精度。

2.4.2原理

互斥特征捆绑算法（Exclusive Feature Bundling, EFB）指出如果将一些特征进行融合绑定，则可以降低特征数量。这样在构建直方图时的时间复杂度从O(#data* #feature)变为 O(#data * #bundle), 这里#bundle指的融合绑定后的特征个数，#bundle << # feature。

针对这种想法，我们会遇到两个问题：

- 怎么判定哪些特征应该绑在一起（build bundled）？
- 怎么把特征绑为一个（merge feature）？

1. 解决哪些特征应该绑在一起

将相互独立的特征进行绑定是一个 NP-Hard 问题，LightGBM的EFB算法将这个问题转化为图着色的问题来求解，将所有的特征视为图的各个顶点，将不是相互独立的特征用一条边连接起来，边的权重就是两个相连接的特征的总冲突值，这样需要绑定的特征就是在图着色问题中要涂上同一种颜色的那些点（特征）。

我们注意到通常有很多特征，尽管不是100%相互排斥，但也很少同时取非零值。 如果我们的算法可以允许一小部分的冲突，我们可以得到更少的特征包，进一步提高计算效率。

具体步骤可以总结如下：

1. 构造一个加权无向图，顶点是特征，边有权重，其权重与两个特征间冲突相关；
2. 根据节点的度进行降序排序，度越大，与其它特征的冲突越大；
3. 遍历每个特征，将它分配给现有特征包，或者新建一个特征包，使得总体冲突最小。

2. 解决怎么把特征绑为一捆

特征合并算法，其关键在于原始特征能从合并的特征中分离出来。绑定几个特征在同一个bundle里需要保证绑定前的原始特征的值可以在bundle中识别，考虑到histogram-based算法将连续的值保存为离散的bins，我们可以使得不同特征的值分到bundle中的不同bin（箱子）中，这可以通过在特征值中加一个偏置常量来解决。

比如，我们在bundle中绑定了两个特征A和B，A特征的原始取值为区间[0,10)，B特征的原始取值为区间[0,20），我们可以在B特征的取值上加一个偏置常量10，将其取值范围变为[10,30），绑定后的特征取值范围为 [0, 30），这样就可以放心的融合特征A和B了。