6. LGB（二） 特点

3. LightGBM的工程优化

3.1 直接支持类别特征

实际上大多数机器学习工具都无法直接支持类别特征，一般需要把类别特征，通过 one-hot 编码，转化到多维的0/1特征，降低了空间和时间的效率。

但我们知道对于决策树来说并不推荐使用 one-hot 编码，尤其当类别特征中类别个数很多的情况下，会产生样本切分不平衡问题，导致切分增益非常小（即浪费了这个特征）

LightGBM优化了对类别特征的支持，可以直接输入类别特征，不需要额外的0/1展开。

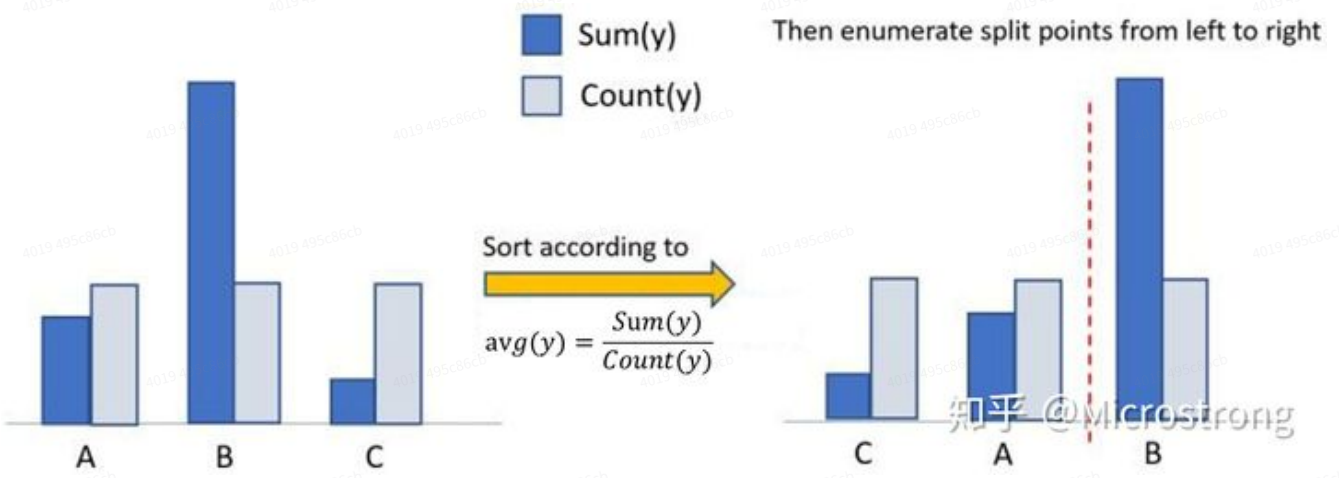
LightGBM采用 many-vs-many 的切分方式将类别特征分为两个子集，实现类别特征的最优切分。实现了O(k logk)的时间复杂度，k是某维度类别数量。

3.1.1算法实现（many-vs-many 的切分方式）

编码方式类似于：目标编码（即把类别 对应的标签的平均值 当成对应的特征值）。

在枚举分割点之前，先把直方图按照每个类别对应的label均值进行排序；然后按照排序的结果依次枚举最优分割点。流程如下图：

然后根据这个替换的值进行最优分裂，将样本分成多个子集。



当然，这个方法很容易过拟合，所以LightGBM里面还增加了很多对于这个方法的约束和正则化。在Expo数据集上的实验结果表明，相比0/1展开的方法，使用LightGBM支持的类别特征可以使训练速度加速8倍，并且精度一致。更重要的是，LightGBM是第一个直接支持类别特征的GBDT工具。

3.2 支持高效并行

3.2.1特征并行

特征并行的主要思想是不同机器在不同的特征集合上分别寻找最优的分割点，然后在机器间同步最优的分割点。XGBoost使用的就是这种特征并行方法。

这种特征并行方法有个很大的缺点：就是对数据进行垂直划分，每台机器所含数据不同，然后使用不同机器找到不同特征的最优分裂点，划分结果需要通过通信告知每台机器，增加了额外的复杂度。

LightGBM 则不进行数据垂直划分，而是在**每台机器上保存全部训练数据**，在得到最佳划分方案后可在本地执行划分而减少了不必要的通信。

3.2.2 数据并行

传统的数据并行策略主要为水平划分数据，然后本地构建直方图并整合成全局直方图，最后在全局直方图中找出最佳划分点。

这种数据划分有一个很大的缺点：通讯开销过大。如果使用点对点通信，一台机器的通讯开销大约为 $O(\#machine * \#feature * \#bin)$ ；如果使用集成的通信，则通讯开销为 $O(2 * \#feature * \#bin)$ ，

LightGBM 采用分散规约（Reduce scatter）的方式将直方图整合的任务分摊到不同机器上，从而降低通信代价，并通过直方图做差进一步降低不同机器间的通信。

3.2.3 投票并行

针对**数据量特别大特征也特别多的情况下**，可以采用投票并行。投票并行主要针对数据并行时数据合并的通信代价比较大的瓶颈进行优化，其通过投票的方式只合并部分特征的直方图从而达到降低通信量的目的。

大致步骤为两步：

1. 本地找出 Top K 特征，并基于投票筛选出可能是最优分割点的特征；
2. 合并时只合并每个机器选出来的特征

4. LightGBM的优缺点

4.1 优点

这部分主要总结下 LightGBM 相对于 XGBoost 的优点，从内存和速度两方面进行介绍。

（1）速度更快

- LightGBM **采用了直方图算法**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 采用了基于 **Leaf-wise 算法**的增长策略构建树，减少了很多不必要的计算量；
- 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；
- LightGBM 对**缓存**也进行了优化，增加了缓存命中率；

（2）内存更小

- XGBoost使用预排序后需要记录特征值及其对应样本的统计值的索引，而 LightGBM 使用了直方图算法将特征值转变为 bin 值，且不需要记录特征到样本的索引，将空间复杂度从 O(2 * # data)

降低为 O(#bin)，极大的减少了内存消耗；

- LightGBM 采用了**直方图算法**将存储特征值转变为存储 bin 值，降低了内存消耗；
- LightGBM 在训练过程中采用**互斥特征捆绑**算法减少了特征数量，降低了内存消耗。

4.2 缺点

- **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限制，在保证高效率的同时防止过拟合；
- Boosting族是迭代算法，每一次迭代都根据上一次迭代的预测结果对样本进行权重调整，所以随着迭代不断进行，误差会越来越小，模型的偏差（bias）会不断降低。由于LightGBM是基于偏差的算法，所以会**对噪点较为敏感**；
- 在寻找最优解时，依据的是最优切分变量，没有将**最优解是全部特征的综合这一理念考虑进去**；