1.阅读背景

集成学习的 GBDT 算法在性能提升上的两种算法：XGBoost 和 LightGBM

2.文献资料

（1）Chen, T., C. Guestrin, and M. Assoc Comp, *XGBoost: A Scalable Tree Boosting System*. Kdd'16: Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining. 2016. 785-794.

（2）Ke, G., et al., *LightGBM: A Highly Efficient Gradient Boosting Decision Tree*, in *Advances in Neural Information Processing Systems 30*, I. Guyon, et al., Editors. 2017.

3.XGBoost

3.1 优点

（1）XGBoost 适用于多个平台，是一个可扩展的系统
（2）提供最高水平的的结果
（3）使用最少数量的资源处理真实世界的问题
（4）提供加权分类和排序方法，包括用户自定义对象
（5）新的稀疏处理方法
（6）近似算法的理论上合理的加权分位数示意图

3.2 技术创新

3.2.1 正则化的学习目标

加入正则化目标来防止过拟合
XGBoost 的目标函数：

$$\min L(\emptyset) = \sum_i \ell(y_i', y_i) + \sum_k \Omega(f_k)$$

$$\text{Where } \Omega(f_k) = \gamma T + \frac{1}{2}\lambda||w||^2$$

3.2.2 拆分查找算法

（1） 基本的贪婪提取算法：数据需要先排序，访问数据时需要有序以便于积累梯度统计

量（在 scikit-learn,单机 XGBoost 里面，单机提升树也会用到）；
优点：算法很强大（列举了所有可能的拆分方式）；
缺点：当数据大小和内存不匹配时，效率并不是很高；在分布式系统并不适用

(2) 近似算法：类似于推荐的思想，先用特征所占的百分比来作为拆分候选点的参考，再用合计统计量来评估出最优方案。
两个变量：全局变量与局部变量
a) 全局变量在树构造的初始阶段起推荐作用；局部变量在每次划分后重新给出推荐建议；
b) 全局方法比局部方法相比需要较少的推荐步数，但需要更多的候选点（因为局部方法会在划分后精炼候选点，但全局方法没有这样做）；
c) 局部方法对深度较大的树更加适合；
d) 在给定足够多的候选点的情况下，全局方法能达到与局部方法一样的精确度

(3) 加权分位数算法：在传统算法中，当每个实例有相同的权重时，分位数示意图算法可以解决这个问题，但是却没有算法能解决加权了的数据集
XGBoost 给出了一个能在加权数据上的算法，并支持一定合并和剪枝操作（这些操作能维持一定的精度水平）

(4) 稀疏觉知的拆分查找：因为在实际情况中，数据很可能稀疏（缺失值；大量的零；人工增加的特征如 one-hot 编码）
一般的方法是从数据中学习最优的默认分类方向
XGBoost 提供各种稀疏方案，时间复杂度为非缺失数据的线性复杂度，稀疏觉知的算法比传统的算法快 50 倍

3.3 系统设计

(1) 并行学习的块列：（优化拆分查找的计算复杂性）
存在一个内存单元的数据称为块；数据在每一块中以压缩列的形式存储，并以列相关特征值排序；
在贪婪提取算法中，将所有数据存储在一个块中，一次块扫描就能获得所有叶子分支中的候选划分的统计量
在近似算法中：多个块被使用，每个块对应数据集中的由行组成的子集；
不同的块既可以分布在不同的机器上，也可以在磁盘核外设置上排序；
分位数查找变成了线性扫描在排序了的列上；在本地推荐算法中，也有作用（候选点在每个分支频繁产生）；直方图聚合的二分查找也变成了线性时间；
划分查找算法时间复杂度大大降低

(2) 缓存感知访问：
具体地，给每个线程分配内部缓存，把梯度统计量放入内存，以小批量的方式执行累加操作。
在数据集大的情况下，贪婪提取算法的缓存感知实现比原始版本快一倍
在近似算法中，选择合适的块大小，为块中能容纳的最大的样本数。

(3) 堆外机算块：
减少读开销和提高磁盘 IO 吞吐量
使用块压缩和块分片来提高堆外计算性能

3.4 总结

XGBoost 合并了正则化目标来防止过拟合；列采样；稀疏感知；贪婪提取算法；近似算法；缓存感知；堆外计算；加权分位数示意图；

4.LightGBM

4.1 优点

其他方法在处理高维及大数量数据是比较耗时，LightGBM 能大大提升处理高维数据的性能

4.2 技术创新

4.2.1GOSS:Gradient-based One-Side Sampling

（1）思想：在下采样时，保留有大梯度的实例，随机丢掉小梯度数据（比随机采样的精确度更高）
（2）难点：简单地丢弃小梯度样本会改变样本分布，会最终影响学习模型的精度，GOSS 能避免这个问题；GOSS 保留所有大梯度数据，在小梯度数据上随机采样；
对小梯度数据引入固定乘子。具体过程如下：
根据数据梯度的绝对值将数据排序，选择 a*100%的样本，在剩余样本中随机采样出 b*100%个样本，在计算信息增益时，通过乘以$\frac{1-a}{b}$放大小梯度数据的信息增益,这样就可以不改变数据的分布。

4.2.2EFB:Exclusive Feature Bunding

（1）思想：稀疏数据的存在，很多特征是专用的（几乎不同时使用非零，如 one-hot 编码）合并类似的专用特征来减少特征数量，问题的解是 NP-hard，因此使用贪心算法能达到一定的近似（按照有权度来为图中所有的点排序，然后把特征合并到度小于某个阈值的社团中或单独创建一个社团。）
（2）具体过程：

1. 判断哪些特征能绑定在一起
2. 如何构造合并束

如果把特征抽象成图中的点，特征之间的冲突看作是图中的边，那么问题就转换为找出图中的社团并使图中的社团数量最少。LGB 里提出了一个贪心的策略，按照有权度来为图中所有的点排序，然后把特征合并到度小于某个阈值的社团中或单独创建一个社团。
对于特征如何合并，一个重要的原则就是使合并的两个特征可以被顺利区分出来，LightGBM

采取了一个更改阈值的方法。例如对于特征 $x \in (0, 10)$， 特征 $y \in (0, 20)$，就可以把特征 y 转换为 $y \in (10,30)$，然后再去合并 x 与 y。

4.3 总结

LightGBM 在处理高维度数据时，在保持精度的水平下，是最快的，即耗时少（比 XGBoost 更快）