LightGBM和XGBoost对比：

- 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

为什么做LightGBM?

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。

XGBoost原理：

- 首先，对所有特征都按照特征的数值进行预排序。
- 其次，在遍历分割点的时候用O(#data)的代价找到一个特征上的最好分割点。
- 最后，找到一个特征的分割点后，将数据分裂成左右子节点。

优点：

能找到最佳分割点

缺点：

首先，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（例如排序后的索引，为了后续快速的计算分割点），这里需要消耗训练数据两倍的内存。

其次，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。

最后，对 cache 优化不友好。在预排序后，特征对梯度的访问是一种随机访问，并且不同的特征访问的顺序不一样，无法对 cache 进行优化。同时，在每一层长树的时候，需要随机访问一个行索引到叶子索引的数组，并且不同特征访问的顺序也不一样，也会造成较大的 cache miss。

LightGBM优化：

Histogram算法（直方图）：先把连续的浮点特征值离散化成k个整数，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。

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)。

缺点：不能找到精确得分割点，但是实际情况无伤大雅

构建直方图用了一个很巧妙地方式：一个叶子节点得直方图由父节点减去另一个叶子节点得到

Leaf-Wise：

正常的leave-wise是每层都是完美二叉树

leaf-wise不是，只找出分裂信息增益最大得一个叶子，分裂，同等次数下，降低更多错误，更好的精度

缺点可能产生更深得决策树，所以加上了深度限制，保证高效防止过拟合

支持类别特征：

不用将类别特征转化成0/1特征，加快效率

并行优化：

lightGBM 原生支持并行学习，目前支持**特征并行**和**数据并行**的两种。

- 特征并行的主要思想是在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。
- 数据并行则是让不同的机器先在本地构造直方图，然后进行全局的合并，最后在合并的直方图上面寻找最优分割点。

LightGBM 针对这两种并行方法都做了优化：

在特征并行算法中，通过在本地保存全部数据避免对数据切分结果的通信；

在数据并行中使用分散规约 (Reduce scatter) 把直方图合并的任务分摊到不同的机器，降低通信和计算，并利用直方图做差，进一步减少了一半的通信量。基于投票的数据并行则进一步优化数据并行中的通信代价，使通信代价变成常数级别。在数据量很大的时候，使用投票并行可以得到非常好的加速效果。

1其他注意

- 当生长相同的叶子时，Leaf-wise 比 level-wise 减少更多的损失。
- 高速，高效处理大数据，运行时需要更低的内存，支持 GPU
- 不要在少量数据上使用，会过拟合，建议 10,000+ 行记录时使用。