**GBDT算法理解**

Boosting方法是一种集成学习方法，试图将多个基学习器组成一个性能更强的学习器。不同于Bagging方法，Boosting方法通过分步迭代（stage-wise）的方式来构建模型，在迭代的每一步构建的弱学习器都是为了弥补已有模型的不足。

GBDT算法基于Boosting集成方法的思想，利用损失函数的负梯度在当前模型的值来作为之前残差的近似值，通过拟合上一轮模型训练的残差，以达到降低偏差的目的。

1. **算法结构**

梯度提升树学习方法的三要素为模型（加法模型），策略（损失函数）和算法（前向分步算法）。

类似于其他的Boosting类算法，GBDT算法可以看是*K*棵树组成的加法模型：



其中为所有决策树组成的空间，代表第*k*轮训练所生成的决策树模型。

对于GBDT而言，类似于其他机器学习算法，其目标函数可以定义为如下：



*l*表示的是loss函数，其损失函数可根据要解决的任务进行选择。

与一般的机器学习算法不同的是，算法直接学习函数（决策树）集合。因为学习的是加法模型，如果能够从前往后，每一步只学习一个基函数及其系数（结构），逐步逼近优化目标函数，那么就可以简化复杂度。

GBDT算法以前一轮模型预测与真实值的残差为为学习目标，使用损失函数的负梯度在当前模型的值来替代残差，不断迭代进行学习。在前向分布算法的第*m*步，给定当前模型，则生成的第*m*个决策模型的模型参数可由下式求得



综上所述，总结一下GBDT算法结构：

1. 初始化得到一个基函数。
2. 计算损失函数在每个训练样本点的负梯度来近似残差。
3. 该轮以计算得到的负梯度值作为训练的目标，生成新的决策树。
4. 把新生成的决策树和原先的模型相加。
5. 当训练误差小于一定范围或者已达到预先指定模型个数，返回模型，否则继续执行(2)-(4)。
6. **优化方法**
   1. **加入正则项**

在代价函数里加入了正则项，控制模型复杂度，一定程度上避免过拟合现象。



正则项里包含了树的叶节点个数、每个叶子节点上输出的score的L2模的平方和。从bias-variance tradeoff角度来讲，正则项降低了模型variance，使学习出来的模型更加简单，防止过拟合，这也是XGBoost优于传统GBDT的一个特性。

* 1. **二阶泰勒展开**

传统GBDT在优化时只用到一阶导数信息（负梯度），XGBoost则对代价函数进行了二阶泰勒展开，同时用到一阶和二阶导数。且XGBoost工具支持自定义代价函数，只要函数可一阶和二阶求导。



* 1. **自动处理缺失值**

对于特征的值有缺失的样本，可以自动学习出它的分裂方向。利用稀疏感知算（Sparsity-Aware Split Finding Algorithm）。本身在节点分裂时不考虑缺失值的数值，缺失值数据被分到左子树和右子树分别计算损失，选择较优的那一个。

* 1. **并行化处理**

并行是在特征粒度上的。决策树的学习最耗时的一个步骤就是对特征的值进行排序（因为要确定最佳分割点），在训练之前，预先对数据进行了排序，然后保存为block结构，后面的迭代中重复地使用这个结构，大大减小计算量。这个block结构也使得并行成为了可能，在进行节点的分裂时，需要计算每个特征的增益，最终选增益最大的那个特征去做分裂，那么各个特征的增益计算就可以开多线程进行。

**2.5 分割点寻找**

树节点在进行分裂时，需要计算每个特征的每个分割点对应的增益，即用贪心法 greedy algorithm枚举所有可能的分割点。当数据无法一次载入内存或者在分布式情况下，贪心算法效率就会变得很低，所以XGBoost还提出了一种可并行的近似approximate algorithm直方图算法Weighted Quantile Sketch，用于高效地生成候选的分割点。

**2.6 自适应学习速率**

在每一步tree boosting之后增加了一个参数（权重），通过这种方式来减小每棵树的影响力，给后面的树提供空间去优化模型。

**2.7 列采样**

借鉴随机森林的方法，防止过拟合的效果比传统的行抽样还好（行抽样功能也有），并且有利于并行化处理算法。

1. **目标函数的选择**

梯度提升是一种用于回归，分类和排序任务的机器学习技术。XGBoost允许自定义目标函数（损失函数），带来了极大的灵活性，可根据问题类型和特点，选择适当的损失函数。

* 1. **回归**

1. 均方差损失，L2损失

简单，计算方便，是一种很好的相似性度量标准，一阶可导，但是对异常点敏感。如果数据中存在异常点，那么误差值就会很大。

1. 平均绝对误差，L1损失

如果异常点代表很重要的异常情况，并且需要被检测出来，则应选用均方差损失函数。相反，如果只把异常值当作受损数据，则应选用平均绝对损失函数。

1. Huber Loss



融合了平方损失和绝对值损失的优点，性能表现优于前两者，当误差较小时，采用均方差损失，当偏差较大时，采用了平均绝对损失，参数控制着误差的临界值。

* 1. **分类**

1. 对数损失函数（交叉熵损失函数）

做二分类问题时，交叉熵函数常常和sigmoid函数一起出现，而进行多分类时，交叉熵函数与softmax函数几乎都一起出现。而且交叉熵损失能够避免误差小时梯度较小的问题, 交叉熵能加速收敛。

1. 合页损失函数

合页损失函数想让正确分类的“得分”比其他错误分类的“得分”高出至少一个边界值, 具有一定的泛化性能，但是合页损失对具体的得分细节不关心。

1. 指数损失函数

指数损失为AdaBoost中使用的损失函数，使用指数损失函数能比较方便地利用加法模型推导出AdaBoost算法。然而，类似于均方差损失函数，对异常点敏感，鲁棒性不好。