GBDT

梯度提升决策树GBDT（Gradient Boosting Decision Tree）也被称为是MART（Multiple Additive Regression Tree)）或者是GBRT（Gradient Boosting Regression Tree），也是一种基于集成思想的决策树模型，但是它和Random Forest有着本质上的区别。

不得不提的是，GBDT是目前竞赛中最为常用的一种机器学习算法，因为它不仅可以适用于多种场景，更难能可贵的是，GBDT有着出众的准确率。这也是为什么很多人称GBDT为机器学习领域的“屠龙刀”。

Boosting，迭代，即通过迭代多棵树来共同决策。这怎么实现呢？

难道是每棵树独立训练一遍，比如A这个人，第一棵树认为是10岁，第二棵树认为是0岁，第三棵树认为是20岁，我们就取平均值10岁做最终结论？

当然不是！且不说这是投票方法并不是GBDT，只要训练集不变，独立训练三次的三棵树必定完全相同，这样做完全没有意义。

之前说过，GBDT是把所有树的结论累加起来做最终结论的，所以可以想到每棵树的结论并不是年龄本身，而是年龄的一个累加量。

GBDT的核心就在于：

每一棵树学的是之前所有树结论和的残差，这个残差就是一个加预测值后能得真实值的累加量。

比如A的真实年龄是18岁，但第一棵树的预测年龄是12岁，差了6岁，即残差为6岁。
那么在第二棵树里我们把A的年龄设为6岁去学习，如果第二棵树真的能把A分到6岁的叶子节点，那累加两棵树的结论就是A的真实年龄
如果第二棵树的结论是5岁，则A仍然存在1岁的残差，第三棵树里A的年龄就变成1岁，继续学
如果我们的迭代轮数还没有完，可以继续迭代下面，每一轮迭代，拟合的岁数误差都会减小。这就是Gradient Boosting在GBDT中的意义。
其实从这里我们可以看出GBDT与Random Forest的本质区别，GBDT不仅仅是简单地运用集成思想，而且它是基于对残差的学习的。我们在这里利用一个GBDT的经典实例进行解释。

以下是GBDT的基本工作原理：

1. **初始化：** 使用一个简单的模型（通常是一个常数，如平均值）来拟合训练数据。

2. **残差计算：** 计算当前模型对训练数据的预测与实际标签之间的残差（误差）。
3. **训练弱学习器：** 构建一棵决策树，目标是拟合当前残差。新树的预测值与前面所有树的预测值相加，以逐步纠正模型的错误。
4. **更新模型：** 将新树的预测值与之前模型的预测值相加，更新模型。
5. **重复：** 重复上述过程，逐步构建更多的树，每次都在之前模型的基础上纠正残差。
6. **生成最终模型：** 最终模型是所有树的累积，即将每棵树的预测值相加。

GBDT的关键是通过梯度下降法（Gradient Descent）来最小化损失函数，不断优化模型的预测性能。由于每个新树都是为了纠正之前模型的错误，因此每棵树都在尝试提高整体模型的性能。

GBDT主要的优点有：

可以灵活处理各种类型的数据，包括连续值和离散值。

在相对少的调参时间情况下，预测的准备率也可以比较高。这个是相对SVM来说的。

使用一些健壮的损失函数，对异常值的鲁棒性非常强。比如 Huber损失函数和Quantile损失函数。

GBDT的主要缺点有：

由于弱学习器之间存在依赖关系，难以并行训练数据。不过可以通过自采样的SGBT来达到部分并行。

梯度提升和梯度下降有什么区别和联系

两者都是在每一轮迭代中，利用损失函数相对于模型的负梯度方向的信息来对当前模型进行更新，只不过在梯度下降中，模型是以参数化的形式表示，从而模型的更新等价于参数的更新。

而在梯度提升中，模型并不需要参数化表示，而是直接定义在函数空间中，从而大大扩展了可以使用的模型种类。

如何理解boosting和bagging

Bagging通过模型集成降低方差，提高弱分类器的性能。

Boosting通过模型集成降低偏差，提高弱分类器的性能。

	Bagging	Boosting
降低	方差	偏差
训练	各个弱分类器可独立训练	弱分类器需要依次生成
典型方法	随机森林	Adaboost，GBDT，XGBoost

训练过程

用每个样本的残差训练下一棵树，直到残差收敛到某个阈值以下，或者树的总数达到某个上限为止。

优缺点

优点

（1）预测阶段计算速度块，树与树之间可并行化计算

（2）在分布稠密的数据集上，泛化能力和表达能力都很好

（3）采用决策树作为弱分类器使得GBDT模型具有较好的可解释性和鲁棒性，能够自动发现特征间的高阶关系，并且不需要对数据进行特殊的预处理如归一化等

缺点

（1）GBDT在高维稀疏的数据集上，表现不如支持向量机或者神经网络

（2）GBDT在处理文本分类特征问题上，优势不如在处理数值特征时明显

（3）训练过程需要串行训练，只能在决策树内容采用一些局部并行手段提高训练速度

GDBT对异常值敏感。对于回归类的问题，如果采用平方损失函数。当出现异常值时，后续模型会对异常值关注过多。

训练过程中有哪些参数对模型的影响比较大

减小步长需要对应增加最大迭代次数

1. **n_estimators**: 也就是弱学习器的最大迭代次数，或者说最大的弱学习器的个数。一般来说n_estimators太小，容易欠拟合，n_estimators太大，又容易过拟合，一般选择一个适中的数值。默认是100。在实际调参的过程中，我们常常将n_estimators和下面介绍的参数learning_rate一起考虑。
2. **learning_rate**: 即每个弱学习器的权重缩减系数νν，也称作步长，在原理篇的正则化章节我们也讲到了，加上了正则化项
3. **subsample**: 即我们在原理篇的正则化章节讲到的子采样，取值为(0,1]。注意这里的子采样和随机森林不一样，随机森林使用的是放回抽样，而这里是不放回抽样。如果取值为1，则全部样本都使用，等于没有使用子采样。如果取值小于1，则只有一部分样本会去做GBDT的决策树拟合。选择小于1的比例可以减少方差，即防止过拟合，但是会增加样本拟合的偏差，因此取值不能太低。推荐在[0.5, 0.8]之间，默认是1.0，即不使用子采样。