6. Gradient Boosting Decision Tree（GBDT）

1.gbdt介绍

1.1gbdt算法原理

模型的公式化：

$$f(x) = f_0(x) + \sum_{t=1}^{T} learningrate * f_t(x)$$

最初的gbdt的设计并不复杂，f0（x）表示初始的基学习器，后面那一项表示后续拟合的所有其它基学习器。

目标函数：

$$Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i)$$

1.2.实例

假设我们原始的label是：10.8 ,5.0, 4.3, 2.5, 8.9，

而我们预测出的label是：10.5, 5.3, 4.4, 2.6, 9.0，

我们计算一下二者的差值，看看当前的模型的预测误差是怎么样的：

0.3,-0.3,-0.1,-0.1,-0.1

既然原始的tree存在误差，那么我们不是可以以原始的特征矩阵为特征，误差作为新的标签来训练一颗新tree，然后直接对二者进行求和，不就能神奇的降低误差了吗？我们上述的计算结果称之为残差，注意，残差的计算方向是——真实标签-预测结果，

而我们前面的计算，实际上 **残差就是**推广来看就是平方**损失函数的负梯度**：

$$L(y, F(x)) = \frac{1}{2}(y - F(X))^2$$

上式为平方损失函数，我们对其进行求导可得：

$$\frac{\partial J}{\partial F(x_i)} = \frac{\partial \sum_i L(y_i, F(x_i))}{\partial F(x_i)} = \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} = F(x_i) - y_i$$

其一阶梯度，取负号就可以得到**负梯度为** yi-f(xi)，也就是我们上面例子里进行的计算。

这样一来，推而广之，分类问题也能很好的理解了，例如对于二分类问题，计算二元交叉熵的负梯度，将负梯度作为下一轮的tree进行拟合时使用的标签。

1.3正则化手段（学习率）

gbdt引入了所谓shrinkage的正则化手段对模型进行约束，说白了就是引入了学习率。

$$f(x) = f_0(x) + \sum_{t=1}^{T} learningrate * f_t(x)$$

即上式中的learningtate，模型之间的组合不再是直接相加，而是乘上学习率

一方面（引入学习率的通用作用）： 引入合适的学习率可以更快更好的收敛到局部或全局最优，学习率太小收敛太慢虽然也能保证达到最优，学习率太大容易发生震荡；

另一方面：学习率的另一个重要作用是（tree独有），增大学习率之后，我们的base tree的数量会增加，实际上就是引入了更多的tree来进行共同决策，比如学习率0.1的时候我们可能100棵tree就stop了，而学习率为0.001的时候就可能需要10000棵tree才stop，这样整个gbdt的决策就是由于更多的base tree来共同支持，预测的稳定性更好，更不容易过拟合。

举个极端的例子，假设gbdt只有一棵base tree，则整个gbdt的稳定性是很差的，完全由这一棵tree来决定，此时退化为普通的决策树，泛化性能大大下降。

2.gbdt的优点

1. 低偏差，拟合能力强.

同时加入了rf的采样的思想后（早期的gbdt没有引入这样的思想），gbdt在方差和偏差方面都有非常好的表现，是目前在各个领域的一种非常常用和流行的算法，尤其是结构化数据的领域，例如典型的风控领域，gbdt可以说是统治了半壁江山。

2. 在训练的过程中自动进行特征选择；

将上游的特征工程和下游的模型训练灵活的结合在一个训练过程中；(嵌入式)

3. 能较好的处理非线性问题；

4. 能够适应多种损失函数；

只要你能将实际的问题抽象成一个有监督问题，并且其损失函数是可导的

3.gbdt的缺点

3.1难以处理高维稀疏的数据

切分增益很小没有太大意义

假设1000万个样本，高维稀疏的情况下，某个稀疏特征的0有9999900个，而1只有100个，此时切分出来左枝的样本有9999900个样本，右枝的样本只有100个，这个时候tree切分的增益是非常小的，不平衡的切分和不切分没什么区别。

3.2对于异常点较为敏感

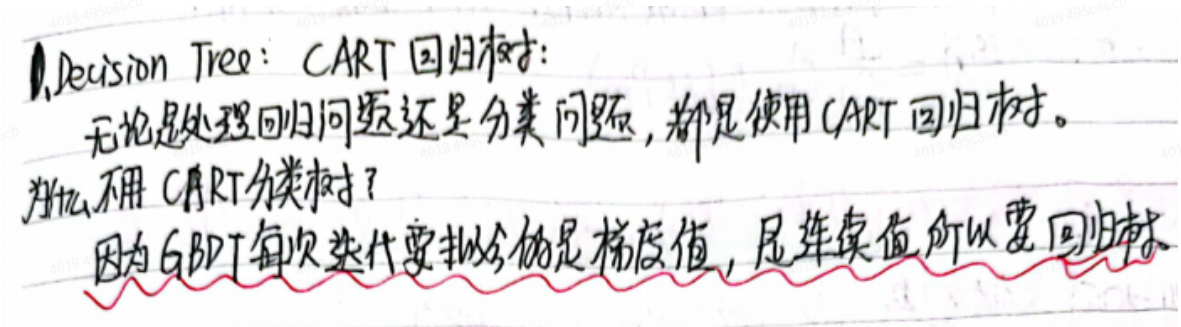
注意是**异常点不是异常值**，对于特征层面异常值，gbdt一点儿也不敏感，因为tree本身是基于特征的相对排序性来分裂的，特征是 1 2 3 4 5 还是 1 2 3 4 100，分裂的时候没有区别.

因为异常点本身比较难有一个好的超平面去拟合，导致了对于这类样本的预测误差往往比较大，gbdt会在异常导致预测误差特别大地样本上不断地去用新的tree来拟合，**导致模型太过拟合异常样本**，最终的结果就是泛化性能.

3.3 集成模型本身的计算复杂度都是比较高的，训练耗时.

4.面试问题

1.为什么GBDT会使用回归树而不是分类树



2.Adaboost与GBDT差异

问题：Adaboost 与 GBDT 的异同：

1) Adaboost 通过改变样本和基学习器权重，来提升模型性能。

2) GBDT：通过训练基学习器拟合模型的残差来……