7. Exterme Gradient Boosting(XGBoost)

XGBoost是Exterme Gradient Boosting（极限梯度提升）的缩写，它是基于决策树的集成机器学习算法，大体上沿袭了之前说过的gbdt的框架，但是在此之上做了很多的改进

1.相对于GBDT的改进

1.1正则化概念的引入

传统的gbdt对模型进行优化的方法就是引入学习率来缓解模型过拟合的问题

xgboost则引入了树的正则化的方法，在原始的代价函数上加入对模型的复杂度的定义作为代价函数的补充项从而生成最终的目标函数，

$$Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

其中：

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

T表示**叶子节点的数量**； gamma则是这一个惩罚项的超参数，人工进行设置；

w表示叶子节点的权重，也就是叶子节点的值 。

第一项：希望叶子节点的数目能够少一点。

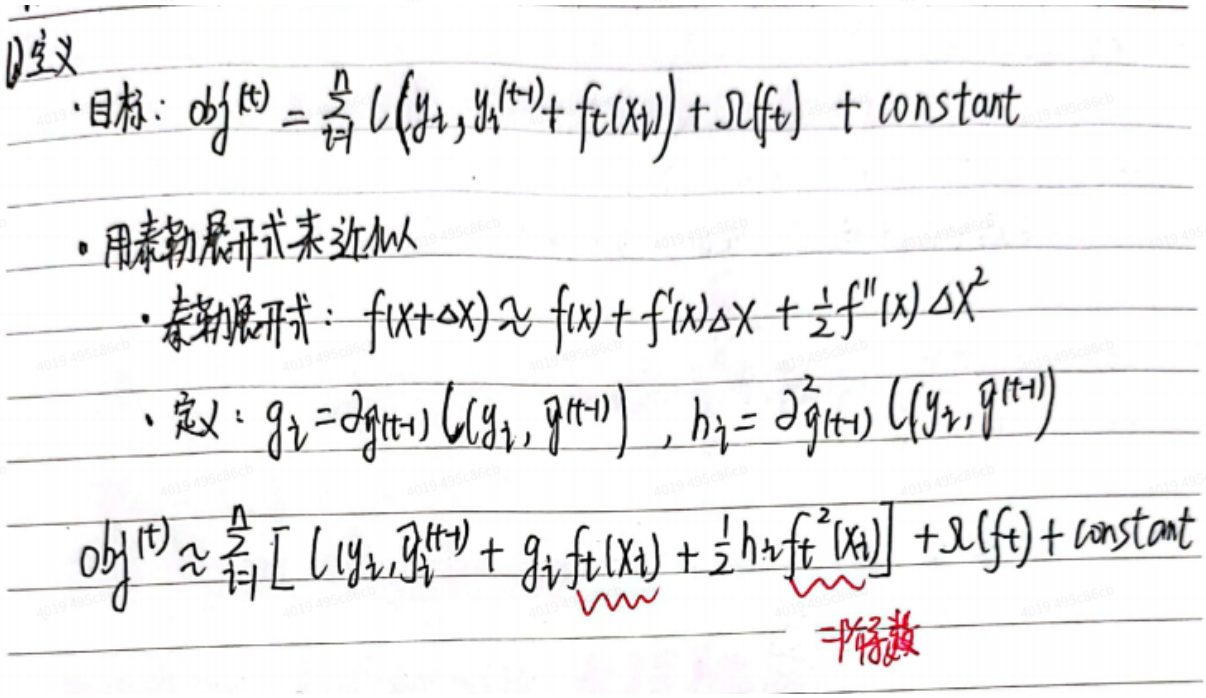
第二项：wi表示叶子节点的权重，和逻辑回归一样，我们希望权重整体是偏小的，因为过大的权重会给预测结果带来很大的不稳定性。

其实抛开前面的gamma*T，后面这项的定义式和l2正则化的形式是完全一样的。

1.2 梯度信息的使用

原始的gbdt仅仅使用了一阶梯度的信息，而xgboost使用了一、二阶梯度。

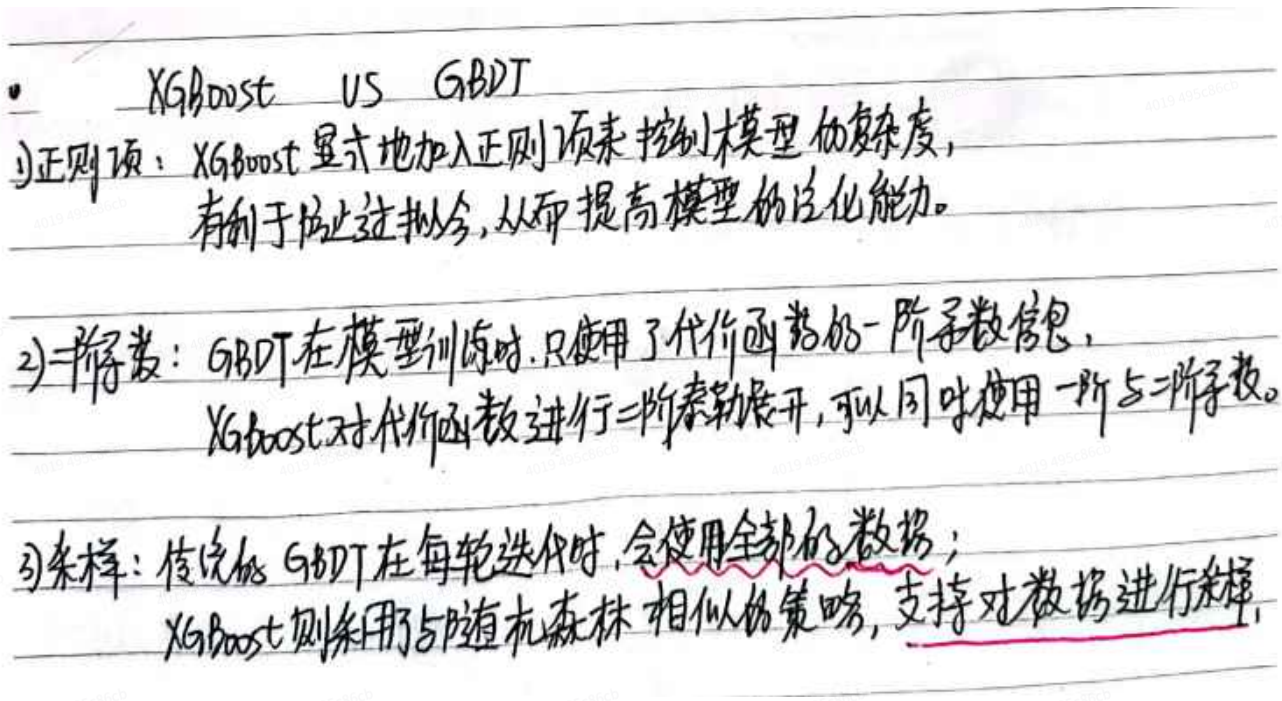
Xgboost对损失函数实际上进行了二阶泰勒展开，这里要注意，我们所说的二阶泰勒展开是针对于gbdt原始的损失函数进行展开的，不包括tree的正则项的部分。



泰勒公式的作用：用多项式函数去逼近光滑函数。

这里xgboost所做的就是直接对原始的损失函数进行二阶泰勒展开，舍弃其误差项，将展开之后的新的损失函数作为模型训练过程中使用的损失函数。

2.XGBoost与GBDT的不同



4）支持线性分类器

传统GBDT以CART作为基分类器，xgboost还支持线性分类器，这个时候xgboost相当于带L1和L2正则化项的逻辑斯蒂回归（分类问题）或者线性回归（回归问题）

3.XGBoost的缺点

1. 难以处理高维稀疏的数据

因为tree本身的正则化对于高维稀疏的数据情况不像l1正则化能够带来有效的约束，并且tree本身的分裂在稀疏的情况下显著性很差；

2. 对于异常点较为敏感

因为gbdt会在因为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差；例如在回归问题中，假设标签是【1.6，2.6，3.5，1.5，1000】，这种情况下gbdt会不断的用新tree去拟合标签为1000的样本的负梯度。

3. 集成模型本身的计算复杂度都是比较高的，训练耗时

4.为什么xgb做二阶泰勒展开不做三阶泰勒或者n阶泰勒展开?

实际上这是精度和工程性能上的一个trade off。

我们知道，当对函数进行n阶展开的时候，n趋近于无穷大的时候，精度也是趋近于无穷大的，但是这也意味着计算量的增大，我们原来只要计算一阶和二阶梯度，如果引入三阶梯度，则又需要额外去计算三阶梯度，如果引入四阶梯度则。。。。，显然，对于gbdt这样动辄成白上千棵tree的算法来说，这无疑大大增加了gbdt的模型训练的复杂度，并且对于gbm的框架来所，单个基学习器的少量误差并不是那么重要，精度不够，tree的数量来凑就好了~