**18122698 方奕丹**

1. Boosting
2. gbdt模型是由k个基模型组成的一个加法运算式：

其中F为所有基模型组成的函数空间

（2）目标函数为：

其中代表了基模型的复杂度，若基模型是树模型，则树的深度、叶子节点数等指标可以反应树的复杂程度。

（3）Boosting是前向优化算法，即从前往后，逐渐建立基模型来优化逼近目标函数：

每一步都要学习一个新模型（需要拟合的模型），且新模型的加入总是以优化目标函数为目的。此时，目标函数可写成

其中

事实上，我们只要求出每一步损失函数的一阶导和二阶导的值带入上式，然后最优化目标函数，就可以得到每一步的。然后根据加法模型得到一个整体模型。

2、用决策树来表示上一步的目标函数

（1）可以转成,代表每个样本落在哪个叶子结点上，而则代表了哪个叶子节点取什么值，所以就代表了每个样本的取值（即预测值）。

（2）决策树模型的复杂度由生成树的叶子节点数量和叶子结点对应的值向量的L2范数决定

（3）假设为第j个叶子节点的样本集合，则

即之前样本的集合，都改写成了叶子结点的集合，由于一个叶子结点有多个样本存在，因此才有了和这两项

定义,则上式可以写成

如果树的结构是固定的，即q是确定的，或者说我们已经知道了每个叶子节点都有哪些样本，所以和是确定的，但不确定（其实就是我们需要预测的值）

那么令目标函数一阶导为0，则可以求得叶子结点j对应的值：

目标函数可以简化为：

3、优化目标函数

* 对于单棵决策树，理想的优化状态为枚举所有可能的树结构，过程如下：

1. 首先枚举所有可能的树结构，即q
2. 计算每种树结构下的目标函数值（2）
3. 取目标函数值最小值为最佳的树结构，根据等式（1）求得每个叶子节点的取值，即样本的预测值。

* 用贪心策略来优化：

1. 从深度为0的树开始，对每个叶节点枚举所有的可用特征
2. 针对每个特征，把属于该节点的训练样本根据该特征值升序排列，通过线性扫描的方式来决定该特征的最佳分裂点，并记录该特征的最大收益（采用最佳分裂点时的收益）
3. 选择收益最大的特征作为分裂特征，用该特征的最佳分裂点作为分裂位置，把该节点生长出左右两个新的叶节点，并为每个新节点关联对应的样本集。
4. 回到第一步，递归执行到满足特定条件为止。

* 计算收益的方法：假设我们在上某一结点上而分裂成两个结点，分别是左（L）和右（R），则分裂前的目标函数是：

分裂后则是

则对于目标函数来说，分裂后的收益是

4、GBDT算法总结：

（1）在算法拟合的每一步都新生成一颗决策树

（2）在拟合这棵树之前，需要计算损失函数在每个样本上的一阶导和二阶导，即和

（3）通过贪心策略生成一棵树，计算每个叶子结点的和，利用（1）计算预测值

（4）把新生成的决策树加入

其中为学习率，主要为了抑制模型的过拟合。