XGBoost算法

XGBoost是boosting算法的其中一种。Boosting算法的思想是将许多弱分类器集成在一起形成一个强分类器。因为XGBoost也是一种梯度提升树模型，与GBDT树模型类似，都是将许多树模型集成在一起，形成一个很强的分类器。所用到的树模型就是**CART**回归树模型。

一、CART回归树

CART回归树假设树为二叉树，通过不断将特征进行分裂。比如当前树结点是基于第j个特征值进行分裂的，设该特征值小于s的样本划分为左子树，大于s的样本划分为右子树。

$$R_1(j,s) = \{x|x^j \leq s\} \quad and \quad R_2(j,s) = \{x|x^j > s\}$$

而CART回归树实质上就是在该特征维度对样本空间进行划分，而这种空间划分的优化是一种NP难问题，因此，在决策树模型中是使用启发式方法解决。典型CART回归树产生的目标函数为：

$$\sum_{x_i \in R_m} (y_i - f(x_i))^2$$

因此，当我们为了求解最优的切分特征j和最优的切分点s，就转化为求解这么一个目标函数：

$$\underset{j,s}{min}[\underset{c_1}{min} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \underset{c_2}{min} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2]$$

所以我们只要遍历所有特征的的所有切分点，就能找到最优的切分特征和切分点。最终得到一棵回归树。

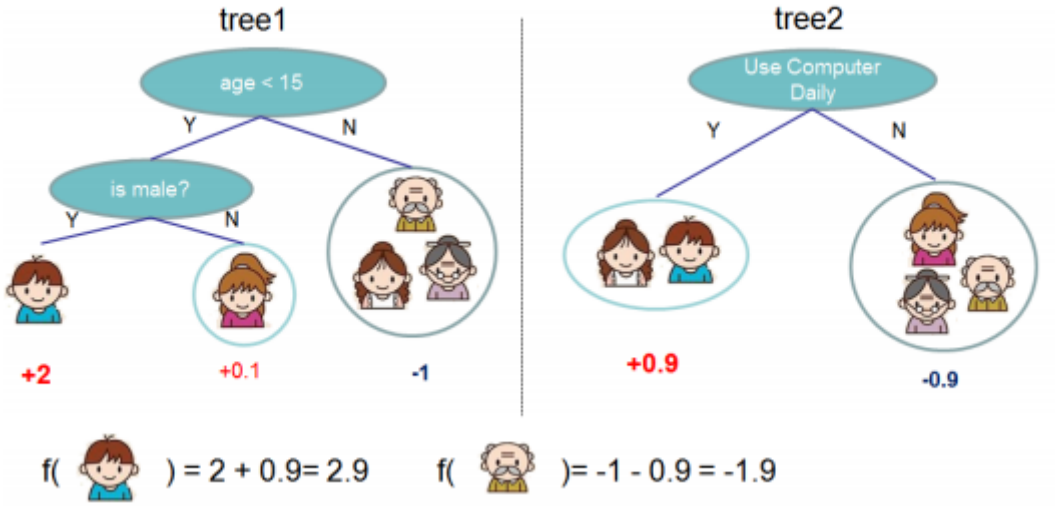
二、XGBoost算法思想

该算法思想就是不断地添加树，不断地进行特征分裂来生长一棵树，每次添加一个树，其实是学习一个新函数，去拟合上次预测的残差。当我们训练完成得到M棵树，我们要预测一个样本的分数，其实就是根据这个样本的特征，在每棵树中会落到对应的一个叶子节点，每个叶子节点就对应一个分数，最后只需要将每棵树对应的分数加起来就是该样本的预测值。

$$\hat{y} = f_m(x) = \sum_{m=1}^{M} f_i(x) \quad s.t. \quad f(x) = w_q(x)$$

w_q(x)为叶子节点q的分数，f(x)为其中一棵回归树.

例如：训练出了2棵决策树，小孩的预测分数就是两棵树中小孩所落到的结点的分数相加。爷爷的预测分数同理。



三、XGBoost原理

XGBoost目标函数定义为：

$$L(y_i, f(x_i)) = \sum_{i=1}^{n} l(y_i, f^{t-1}(x_i) + T(x_i;\gamma)) + \gamma T + \lambda \frac{1}{2} \sum_{m=1}^{M} \omega_j^2$$

目标函数由两部分构成，第一部分用来衡量预测分数和真实分数的差距，另一部分则是正则化项。正则化项同样包含两部分，T表示叶子结点的个数，w表示叶子节点的分数。γ可以控制叶子结点的个数，λ可以控制叶子节点的分数不会过大，防止过拟合。

新生成的树是要拟合上次预测的残差的，即当生成t棵树后，预测分数可以写成：

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$$

同时，可以将目标函数改写成：

$$L^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i{}^{(t-1)} + f_t(x_i)) + \Omega(f_t)$$

明显，我们接下来就是要去找到一个f_t能够最小化目标函数。XGBoost的想法是利用其在f_t=0处的泰勒二阶展开近似它。所以，目标函数近似为：

$$L^{(t)} = \sum_{i=1}^{n} [l(y_i, \hat{y}^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)] + \Omega(f_t)$$

其中g_i为一阶导数，h_i为二阶导数：

$$g_i = \frac{\partial l(y_i, \hat{y}^{(t-1)})}{\partial \hat{y}^{(t-1)}} \qquad h_i = \frac{\partial l^2(y_i, \hat{y}^{(t-1)})}{\partial \hat{y}^{(t-1)}}$$

由于前t-1棵树的预测分数与y的残差对目标函数优化不影响，可以直接去掉。简化目标函数为：

$$L^{(t)} = \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)] + \Omega(f_t)$$

上式是将每个样本的损失函数值加起来，我们知道，每个样本都最终会落到一个叶子结点中，我们可以将所以同一个叶子结点的样本重组起来。

$$Obj^{(t)} = \sum_{i=1}^{n}[g_i f_t(x_i) + \frac{1}{2}h_i f_t(x_i)] + \Omega(f_t)$$

$$= \sum_{i=1}^{n}[g_i f_t(x_i) + \frac{1}{2}h_i f_t(x_i)] + \gamma T + \lambda\frac{1}{2}\sum_{m=1}^{M}\omega_j^2$$

$$= \sum_{m=1}^{M}[(\sum_{i} \in I_i g_i)\omega_j + \frac{1}{2}(\sum_{i} \in I_i h_i + \lambda)\omega_j^2] + \gamma T$$

因此通过上式的改写，我们可以将目标函数改写成关于叶子结点分数w的一个一元二次函数，求解最优的w和目标函数值就变得很简单了，直接使用顶点公式即可。因此，最优的w和目标函数公式为

$$\omega_m^* = -\frac{G_m}{H_m + \lambda} \quad Obj = -\frac{1}{2}\sum_{m=1}^{M}\frac{G_m^2}{H_m + \lambda} + \gamma T$$

四、分裂结点算法

在上面的推导中，我们知道了如果我们一棵树的结构确定了，如何求得每个叶子结点的分数。但我们还没介绍如何确定树结构，即每次特征分裂怎么寻找最佳特征，怎么寻找最佳分裂点。

正如上文说到，基于空间切分去构造一颗决策树是一个NP难问题，我们不可能去遍历所有树结构，因此，XGBoost使用了和CART回归树一样的想法，利用贪婪算法，遍历所有特征的所有特征划分点，不同的是使用上式目标函数值作为评价函数。具体做法就是分裂后的目标函数值比单子叶子节点的目标函数的增益，同时为了限制树生长过深，还加了个阈值，只有当增益大于该阈值才进行分裂。

同时可以设置树的最大深度、当样本权重和小于设定阈值时停止生长去防止过拟合。

五、Shrinkage and Column Subsampling

XGBoost还提出了两种防止过拟合的方法：Shrinkage and Column Subsampling。Shrinkage方法就是在每次迭代中对树的每个叶子结点的分数乘上一个缩减权重η，这可以使得每一棵树的影响力不会太大，留下更大的空间给后面生成的树去优化模型。Column Subsampling类似于随机森林中的选取部分特征进行建树。其可分为两种，一种是按层随机采样，在对同一层内每个结点分裂之前，先随机选择一部分特征，然后只需要遍历这部分的特征，来确定最优的分割点。另一种是随机选择特征，则建树前随机选择一部分特征然后分裂就只遍历这些特征。一般情况下前者效果更好。

六、近似算法

对于连续型特征值，当样本数量非常大，该特征取值过多时，遍历所有取值会花费很多时间，且容易过拟合。因此XGBoost思想是对特征进行分桶，即找到有限个划分点，将位于相邻分位点之间的样本分在一个桶中。在遍历该特征的时候，只需要遍历各个分位点，从而计算最优划分。从算法伪代码中该流程还可以分为两种：

- 全局的近似是在新生成一棵树之前就对各个特征计算分位点并划分样本，之后在每次分裂过程中都采用近似划分；
- 局部近似就是在具体的某一次分裂节点的过程中采用近似算法。

七、针对稀疏数据的算法（缺失值处理）

当样本的第i个特征值缺失时，无法利用该特征进行划分时，XGBoost的想法是将该样本分别划分到左结点和右结点，然后计算其增益，哪个大就划分到哪边。

八、XGBoost的优点

之所以XGBoost可以成为机器学习的大杀器，广泛用于数据科学竞赛和工业界，是因为它有许多优点：

1.使用许多策略去防止过拟合，如：正则化项、Shrinkage and Column Subsampling等。

2.目标函数优化利用了损失函数关于待求函数的二阶导数

3.支持并行化，这是XGBoost的闪光点，虽然树与树之间是串行关系，但是同层级节点可并行。具体的对于某个节点，节点内选择最佳分裂点，候选分裂点计算增益用多线程并行。训练速度快。

4.添加了对稀疏数据的处理。

5.交叉验证，early stop，当预测结果已经很好的时候可以提前停止建树，加快训练速度。

6.支持设置样本权重，该权重体现在一阶导数g和二阶导数h，通过调整权重可以去更加关注一些样本。