

## 面试问题

1. 为什么xgboost要用泰勒展开，优势在哪里

2. xgboost如何寻找最优特征？是有放回还是无放回的呢？

3.xgboost怎么给特征评分

1. CART回归树

2. XGBoost算法思想

3.XGBoost原理

4.分裂节点算法-贪心法

5.Shrinkage and Column Subsampling

6.针对稀疏数据的算法（缺失值处理）

7.XGBoost调参

8.XGBoost优点

## 面试问题

### 1. 为什么xgboost要用泰勒展开，优势在哪里

xgboost使用了一阶和二阶偏导，二阶导数有利于梯度下降的更快更准，使用泰勒展开取得函数做自变量的二阶导数形式，可以在不选定损失函数具体形式的情况下，仅仅依靠输入数据的值就可以进行叶子分裂优化计算，本质上也就把损失函数的选取和模型优化算法/参数选择分开了。这种去耦合增加了xgboost的适用性，使得它按需选取损失函数，可以用于分类，也可以用于回归。

### 2. xgboost如何寻找最优特征？是有放回还是无放回的呢？

xgboost在训练的过程中给出各个特征的增益评分，最大增益的特征会被选出来作为分裂依据，从而记忆了每个特征对在模型训练时的重要性-从根到叶子中间节点涉及某特征的次数作为该特征重要性排序。

xgboost属于boosting集成学习方法，样本是不放回的，因而每轮计算样本不重复。另一方面，xgboost支持子采样(subsample)，也就是每轮计算可以不使用全部样本，以减少过拟合。进一步地，xgboost还有列采样(colsample\_bytree)，每轮计算按百分比随机采样一部分特征，既提高速度又减少过拟合。

### 3.xgboost怎么给特征评分

在训练过程中，通过Gini指数选择分裂点的特征，一个特征被选中的次数越多，那么该特征评分越高。

---

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是一种提升树模型，所以它是将许多树模型集成在一起，形成一个很强的分类器。而所用到的树模型则是CART回归树模型。讲解其原理前，先讲解一下CART回归树。

## 1. CART回归树

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

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

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，就转换为求解这么一个目标函数：

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

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。

## 2. XGBoost算法思想

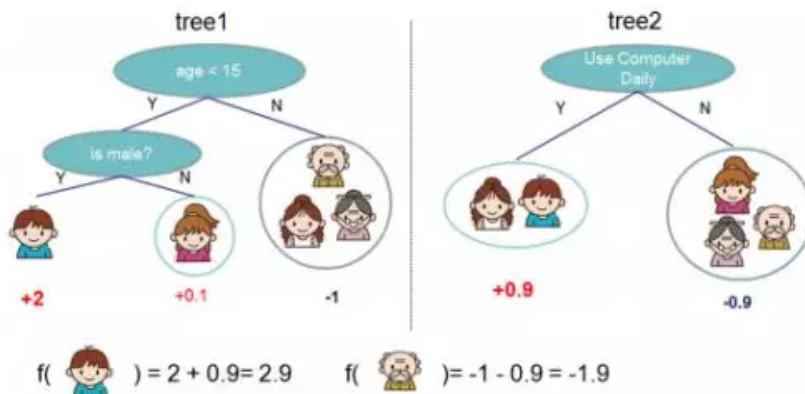
该算法思想就是不断地添加树，不断进行特征分裂来生长一棵树，每次添加一个树，其实是学习一个新函数，去拟合上次预测的残差。当我们训练完成得到k棵树，要预测一个样本的分数，其实就是根据这个样本的特征，在每棵树中落到对应的一个叶子节点，每个叶子节点就对应一个分数，最后只需要将每棵树对应的分数加起来就是该样本的预测值。

$$\hat{y} = \phi(x_i) = \sum_{k=1}^K f_k(x_i)$$

where  $F = \{f(x) = w_{q(x)}\} (q: R^m \rightarrow T, w \in R^T)$

注:  $w_{q(x)}$  为叶子节点q的分数,  $f(x)$  为其中一棵回归树

如下图例子, 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。



### 3.XGBoost原理

XGBoost目标函数定义为:

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

Training loss
Complexity of the Trees

$$\text{where } \Omega(f) = \gamma T + \frac{1}{2} \lambda ||w||^2$$

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。 $\gamma$  可以控制叶子结点的个数,  $\lambda$  可以控制叶子节点的分数不会过大, 防止过拟合。

正如上文所说, 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:

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

很明显，我们接下来就是要找到一个 $f_t$ 能够最小化目标函数，xgboost的想法是利用其在 $f_t=0$ 处的泰勒展开近似它。所以，目标函数近似为

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

其中 $g_i$ 为一阶导数， $h_i$ 为二阶导数

$$g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}), \quad h_i = \partial_{\hat{y}^{(t-1)}}^2 l(y_i, \hat{y}^{(t-1)})$$

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：

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^n [g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t)$$

上式是将每个样本的损失函数加起来，我们知道，每个样本最终都会落到一个叶子节点中，我们可以将同一个叶子节点的样本重组，如下：

$$\begin{aligned} Obj^{(t)} &\simeq \sum_{i=1}^n \left[ g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \\ &= \sum_{i=1}^n \left[ g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2 \right] + \gamma T + \lambda \frac{1}{2} \sum_{j=1}^T w_j^2 \\ &= \sum_{j=1}^T \left[ \left( \sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left( \sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma T \end{aligned}$$

其中 $I$ 被定义为每个叶子上面样本集合

$$I_j = \{i | q(x_i) = j\}$$

定义

$$G_j = \sum_{i \in I_j} g_i \quad H_j = \sum_{i \in I_j} h_i$$

最终公式可以简化为

$$\begin{aligned} Obj^{(t)} &= \sum_{j=1}^T \left[ \left( \sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left( \sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma T \\ &= \sum_{j=1}^T \left[ G_j w_j + \frac{1}{2} (H_j + \lambda) w_j^2 \right] + \gamma T \end{aligned}$$

通过对 $w_j$ 求导等于0，得到






$$w_j^* = -\frac{G_j}{H_j + \lambda}$$

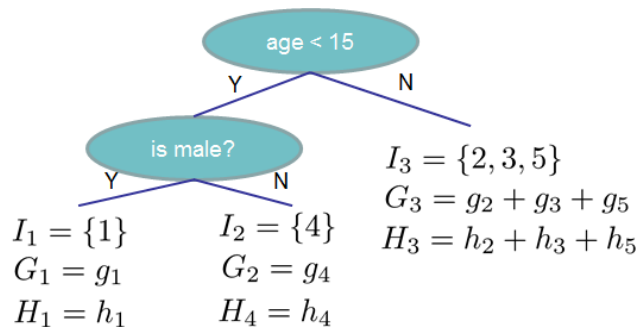
然后把上式代入得到：

$$Obj = -\frac{1}{2} \sum_{j=1}^T \frac{G_j^2}{H_j + \lambda} + \gamma T$$

示例

$Obj$ 代表了当我们指定一个树的结构的时候，我们在目标上面最多减少多少。我们可以把它叫做结构分数(structure score)。

样本号	梯度数据
1 	$g_1, h_1$
2 	$g_2, h_2$
3 	$g_3, h_3$
4 	$g_4, h_4$
5 	$g_5, h_5$



$$Obj = -\frac{1}{2} \sum_j \frac{G_j^2}{H_j + \lambda} + 3\gamma$$

这个分数越小，代表这个树的结构越好

## 4.分裂节点算法-贪心法

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。

贪心法：每一次尝试去对已有的叶子加入一个分割

$$Gain = \underbrace{\frac{1}{2} \left[ \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} \right]}_{\text{左子树分数} + \text{右子树分数}} - \underbrace{\frac{(G_L + G_R)^2}{H_L + H_R + \lambda}}_{\text{不分割我们可以拿到的分数}} - \underbrace{\gamma}_{\text{加入新叶子节点引入的复杂度代价}}$$

观察这个目标函数，大家会发现第二个值得注意的事情就是引入分割不一定会使得情况变好，因为我们有一个引入新叶子的惩罚项。优化这个目标对应了树的剪枝，当引入的分割带来的增益小于一个阈值的时候，我们可以剪掉这个分割。大家可以发现，当我们正式地推导目标的时候，像计算分数和剪枝这样的策略都会自然地出现，而不再是一种因为heuristic（启发式）而进行的操作了。

## 5. Shrinkage and Column Subsampling

XGBoost还提出了两种过拟合的方法：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  $\eta$ ，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。

## 6.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。

## 7.XGBoost调参

看这篇，一定要看！

交叉验证+GridSearch

<https://cloud.tencent.com/developer/article/1080593>

[https://scikit-](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html)

[learn.org/stable/modules/generated/sklearn.model\\_selection.GridSearchCV.html](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html)

<http://www.cnblogs.com/mfryf/p/6293814.html>

## 8.XGBoost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。