19. XGBoost常见问题★

1．简单介绍一下XGBoost

首先需要说一说GBDT，它是一种基于boosting**增强策略的加法模型**，训练的时候采用**前向分布算法**进行**贪婪学习**，每次迭代都学习一棵CART树来拟合之前t-1棵树的预测结果与训练样本真实值的残差。XGBoost对GBDT进行了一系列的优化，比如损失函数进行了**二阶泰勒展开**，目标函数加入了**正则项**，**支持并行**和**默认缺失值处理**等等，在可扩展性和训练速度上有了很大的提升，但其核心思想并没有大的变化。XGBoost是GBDT的工程实现。

2．XGBoost和GBDT有什么区别

- **基分类器**：XGBoost的基分类器不仅支持CART决策树，还支持线性分类器，此时XGBoost相当于带L1和L2正则化项的逻辑回归(分类问题)或者线性回归(回归问题)
- **导数信息**:XGBoost对损失函数做了二阶泰勒展开，GBDT只用了一阶导数信息。并且在损失函数一阶，二阶可导的条件下，XGBoost可以自定义损失函数。
- **正则项**：XGBoost的目标函数加了正则项，相当于预剪枝，使得学习出来的模型更加不容易过拟合。
- **列抽样**：XGBoost支持列抽样，与随机森林类似，用于防止过拟合。
- **缺失值处理**：对树中的每个非叶子节点，XGBoost可以自动学习出它的默认分裂方向。如果某个样本该特征值确缺失，会将其划入默认分支。
- **并行化**：XGBoost 的并行化并不是指的tree维度的并行，而是特征维度的并行，XGBoost预先将每个特征按特征值排好序，存储为块(block)结构，分裂节点时可以采用多线程并行查找每个特征的最佳分割点，极大提升训练速度。这个块存储结构也使得并行成为可能，在进行节点分裂时，需要计算每个特征的信息增益，最终选增益最大的那个特征去做分裂，那么各个特征的增益计算就可以开多线程并行。

	GBDT	XGBoost
定义	是ML的某种算法	是GBDT算法的工程实现
正则项	无	显式的加入，控制过拟合
泰勒展开	损失函数的一阶信息	损失函数的二阶信息
基础分类器	CART	多种分类器
数据使用	使用全部数据	支持采样col_sample和subsample
缺失值	未进行处理	自动学习
并行化	不支持并行	特征维度并行，并行查找每个特征的最佳分割点
控制过拟合		支持列抽样,样本采样,正则项

3．XGBoost和Light GBM的区别

4．XGBoost为什么用二阶泰勒展开

- **精准性**：相对于GBDT的一阶泰勒展开，XGBoost采用二阶泰勒展开，可以更为精准的逼近真实的损失函数。
- **可扩展性**：损失函数支持自定义，只需要新的损失函数二阶可导

5．XGBoost为什么可以并行训练

XGBoost的并行，并非每棵树可以并行训练，XGB本质上还是采用boosting思想，每棵树训练前需要等前面的树训练完成才能开始训练。XGBoost的并行，指的使**特征维度上的并行**，在训练之前，每个特征按照特征值对样本进行预排序，并存储为**block结构**，在后面查找特征分割点时可以重复使用，而且特征已经被存储为一个个block结构，那么**在寻找每个特征的最佳分裂点时**，可以利用多线程对每个block并行计算。

6．XGBoost为什么快

- **分块并行**：训练前每个特征按照特征值排序并存储为Block结构，后面查找特征分割点时重复使用，并且支持并行查找每个特征的分割点。
- **候选分位点**：每个特征采用常数个分位点作为候选分割点。
- **CPU cache 命中优化**：使用缓存预取的方法，对每个线程分配一个连续的buffer，读取每个block中样本的梯度信息并存入连续的Buffer中。
- **Block处理优化**：Block预先放入内存，Block按列进行解压缩，将Block划分到不同硬盘来提高吞吐。

7．XGBoost防止过拟合的方法

- **目标函数添加正则项**：叶子节点个数+叶子节点权重的L2正则化
- **列抽样**：训练的时候只用一部分特征(不考虑剩余的Block块即可)
- **子采样**：每轮计算可以不适用全部样本，使得算法更加保守
- **shrinkage**：学习率/步长,为了给后面的训练留出更多的学习空间。

8．XGBoost如何处理缺失值

- 在特征K上寻找最佳split point时，不会对该列特征missing的样本进行遍历，而只对该列特征值为non-missing的样本上对应的特征值进行遍历，通过这个技巧来减少了为稀疏离

散特征寻找split point的时间开销

- 在逻辑实现上，为了保证完备性，会将该特征值missing的样本分别分配到左叶子结点和右叶子结点，两种情形都计算一遍后，选择分裂后增益最大的那个方向(左分支或者右分支)，作为预测时特征值缺失样本的默认分支方向
- 如果在训练中没有缺失值而在预测中出现缺失，那么会自定将缺失值的划分方向放到**右子结点**

9．XGBoost中叶子结点的权重如何计算出来

XGBoost的目标函数最终推到形式如下:

$$Obj^{(t)} = \sum_{j=1}^{T}[G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2] + \gamma T$$

利用一元二次函数求最值的知识，当目标函数达到最小值Obj*时，每个叶子结点的权重为wj*

具体公式如下

$$w_j^* = -\frac{G_j}{H_j + \lambda}$$ 每个叶子结点的权重

$$Obj = -\frac{1}{2}\sum_{j=1} T\frac{G_j^2}{H_j + \lambda} + \gamma T$$ 第t颗树带来的最小损失(训练损失 + 正则损失)

10．XGBoost中的一颗树停止生长条件

- 当新引入一次分裂所带来的**增益Gain<0**时，放弃当前的分裂。这是训练损失和模型结构复杂度的博弈过程
- 当树**达到最大深度**时，停止建树，因为树的深度太深容易出现过拟合现象，这里需要设置一个超参数，max_depth
- 当引入一次分裂后，重新计算新生成的左右两个叶子结点的样本权重和。如果任一个叶子结点的样本权重低于某一个阈值，也会放弃此次分裂。这涉及到一个超参数：**最小样本权重和**，是指如果一个叶子结点包含的样本数量太少也会放弃分裂，防止树分的太细。

11．XGBoost如何处理不平衡数据

- 在意AUC，采用AUC评估模型的性能，可以通过scale_pos_weight来平衡正样本和负样本的权重
- 在意概率(预测得分的合理性),不能重新平衡数据集(会破坏数据的真实分布),应该设置max_delta_step为一个有限数字来帮助收敛(基模型为LR时有效)
- 通过上采样，下采样，SMOTE算法或者自定义损失函数的方式解决正负样本不平衡的问题。

12．XGBoost中如何对树进行剪枝

- 在目标函数中增加了**正则项**：使用**叶子结点的数目**和**叶子结点权重**的L2模的平方，控制树的复杂度
- 在结点分裂时，定义了一个**阈值**，如果分裂后目标函数的增益小于该阈值，则不分裂。
- 当引入一次分裂后，重新计算新生成的左右两个叶子结点的样本权重和。如果任一个叶子结点的样本权重低于某一个阈值(最小样本权重和),也会放弃此次分裂。
- XGBoost先从顶到底建立树直到最大深度，再从底到顶反向检查是否有不满足分裂条件的结点，进行剪枝。

13．XGBoost怎么选择最佳分裂点

XGBoost在训练前预先将特征按照特征值进行排序，并存储为Block结构，以后在结点分裂时可以重复使用该结构。因此，可以采用**特征并行**的方法利用多个线程分别计算每个特征的最佳分裂点，根据每次分裂后产生的增益，最终选择增益最大的那个特征的特征值作为最佳分裂点。

如果再计算每个特征的最佳分割点时，对每个样本都进行遍历，计算复杂度会很大，这种全局扫描的方法并不适用大数据场景下。XGBoost提供了一种直方图近似算法，对特征排序后仅选择常数个候选分裂位置作为候选分裂点，极大提升了结点分裂时的计算效率。

14．XGBoost的Scalable性如何体现

- 基分类器：弱分类器可以支持CART决策树，也可以支持LR和Linear
- 目标函数：支持自定义loss function 只需要一阶和二阶可导，有这个特性是因为二阶泰勒展开，得到通用的目标函数形式。
- 学习方法：Block结构支持并行化，支持out-of-core计算

15．XGBoost如何评价特征的重要性

- weight 该特征在所有树中被用作分割样本的特征的总次数。
- gain :该特征在其出现过的所有树中产生的平均增益
- cover：该特征在其出现过的所有树中的平均覆盖范围。

注意：覆盖范围是指一个特征用作分割点后，其影响的样本数量，即有多少样本经过该特征分割到两个子结点。

16．XGBoost参数调优的一般步骤

- **确定learning rate 和estimate的数量**
- **max_depth和min_child_weight**

我们调整这两个参数是因为，这两个参数对输出结果的影响很大。我们首先将这两个参数设置为较大的数，然后通过迭代的方式不断修正，缩小范围。

  - max_depth，每棵子树的最大深度，check from range(3,10,2)。
  - min_child_weight，子节点的权重阈值，check from range(1,6,2)。

如果一个结点分裂后，它的所有子节点的权重之和都大于该阈值，该叶子节点才可以划分。

- **gamma** 最小划分损失，min_split_loss (0.1,0.5) 指对于一个叶子节点，当对它采取划分后损失函数的降低值的阈值

- **subsample ， cosample_bytree** (0.6,0.9)
  - subsample 是对训练数据的采样比例
  - cosubsample_bytree 是对特征的采样比例
- **正则化参数**
  - alpha 是L1正则化系数，try 1e-5, 1e-2, 0.1, 1, 100
  - lambda 是L2正则化系数
- **降低学习率** 同时要增加树的数量，通常最后学习率为0.01-0.1

17．为什么XGBoost相比于某些模型对缺失值不敏感

树模型对缺失值的敏感度低，大部分时候可以在数据缺失时使用，原因是，一棵树中每个结点在分裂时，寻找的是某个特征的最佳分裂点(特征值)，完全可以不考虑存在特征值缺失的样本，也就是说，如果某些样本特征值缺失，对寻找最佳分割点的影响不是很大。

18．XGBoost模型如果过拟合了怎么解决

- 直接控制模型的复杂度 max_depth,min_child_weight,gamma 等参数
- 增加随机性，从而使得模型在训练时对于噪声不敏感 subsample 和cosample_bytree
- 直接减小学习率，但同时增加迭代次数 estimator

19．为什么XGBoost泰勒二阶展开后效果就比较好呢?

从为什么会想到引入泰勒二阶的角度来说（**可扩展性**）：XGBoost官网上有说，当目标函数是MSE时，展开是一阶项（残差）+二阶项的形式，而其它目标函数，如logistic loss的展开式就没有这样的形式。为了能有个统一的形式，所以采用泰勒展开来得到二阶项，这样就能把MSE推导的那套直接复用到其它自定义损失函数上。简短来说，就是为了统一损失函数求导的形式以支持自定义损失函数。至于为什么要在形式上与MSE统一？是因为MSE是最普遍且常用的损失函数，而且求导最容易，求导后的形式也十分简单。所以理论上只要损失函数形式与MSE统一了，那就只用推导MSE就好了。

从二阶导本身的性质，也就是从为什么要用泰勒二阶展开的角度来说（**精准性**）：二阶信息本身就能让梯度收敛更快更准确。这一点在优化算法里的牛顿法中已经证实。可以简单认为一阶导指引梯度方向，二阶导指引梯度方向如何变化。简单来说，相对于GBDT的一阶泰勒展开，XGBoost采用二阶泰勒展开，可以更为精准的逼近真实的损失函数。

20．xgboost使用之前是否需要对类别型特征进行one-hot处理

xgboost支持离散类别特征进行onehot编码，**因为xgboost只支持数值型的特征**。但是不提倡对离散值特别多的特征通过one-hot的方式进行处理。因为one-hot进行特征打散的影响，其实是会增加树的深度。针对取值特别多的离散特征，我们可以通过embedding的方式映射成低纬向量。与单热编码相比，实体嵌入不仅减少了内存使用并加速了神经网络，更重要的是通过在嵌入空间中映射彼此接近的相似值，它揭示了分类变量的内在属性。

21．XGBoost需要归一化吗

不需要。首先，归一化是对连续特征来说的。那么连续特征的归一化，起到的主要作用是进行数值缩放。数值缩放的目的是解决梯度下降时，等高线是椭圆导致迭代次数增多的问题。而xgboost等树模型是不能进行梯度下降的，因为树模型是阶越的，不可导。树模型是

通过寻找特征的最优分裂点来完成优化的。由于归一化不会改变分裂点的位置，因此xgboost不需要进行归一化。

- [ ] https://blog.csdn.net/linxid/article/details/80147179 <https://blog.csdn.net/linxid/article/details/80147179>
- [ ] https://blog.csdn.net/karmacode/article/details/102937885 <https://blog.csdn.net/karmacode/article/details/102937885>
- [ ] https://blog.csdn.net/weixin_46649052/article/details/115213618 <https://blog.csdn.net/weixin_46649052/article/details/115213618>