蒋豆芽

机器学习——集成学习 试读

来自 【机器学习面试题汇总与解析（蒋豆芽面试题总结）】 | 66 浏览 | 1 回复 | 2021-06-03

蒋豆芽

+关注

机器学习面试题汇总与解析——集成学习、Adaboost、随机森林、GBDT、xgBoost、LightGBM

1. **LightGBM和xgBoost、GBDT的区别**☆☆☆☆☆
2. **xgBoost和gbdt的区别**☆☆☆☆☆
3. **xgBoost的block结构**☆☆☆☆☆
4. **XGBoost的优缺点**☆☆☆☆☆
5. **集成学习 Bagging Boosting**☆☆☆☆☆
6. **RF和GBDT的区别**☆☆☆☆☆
7. **GBDT是否适合于处理大规模的ID特征**☆☆☆☆☆
8. **LightGBM的直方图 排序后会比xgboost的效果差吗，为什么**☆☆☆☆☆
9. **xgboost正则化项和什么有关**☆☆☆☆☆
10. **随机森林哪两个随机**☆☆☆☆☆
11. **bootstrap怎么做的**☆☆☆☆☆
12. **介绍GBDT的详细计算过程**☆☆☆☆☆
13. **xgb的正则项是什么**☆☆☆☆☆
14. **xgboost缺失值处理方法**☆☆☆☆☆
15. **为什么xgboost要二阶展开?** ☆☆☆☆☆
16. **集成学习的方法有哪些**☆☆☆☆☆
17. **泰勒公式求e的近似值**☆☆☆☆☆
18. **XGBoost 如果损失函数没有二阶导，该怎么办**☆☆☆☆☆
19. **GBDT的G梯度的向量长度为多少**☆☆☆☆☆

==========

- 本专栏适合于**Python已经入门**的学生或人士，有一定的编程基础。

方式。这样才方便背诵

- 如专栏内容有错漏，欢迎在评论区指出或私聊我更改，一起学习，共同进步。
- **相信大家都有着高尚的灵魂，请尊重我的知识产权，未经允许严禁各类机构和个人转载、传阅本专栏的内容。**

==========

关于**机器学习算法**书籍，我强烈推荐一本**《百面机器学习算法工程师带你面试》**，这个就很类似面经，还有讲解，写得比较好。私聊我进群。

关于**深度学习算法**书籍，我强烈推荐一本**《解析神经网络——深度学习实践手册》**，简称CNN book，通俗易懂。私聊我进群。

参考资料

B站机器学习视频：https://space.bilibili.com/10781175/channel/detail?cid=133301

这篇文章不错，该讲的都讲到了：https://blog.csdn.net/perfect1t/article/details/83684995

xgBoost： https://www.cnblogs.com/mantch/p/11164221.html

LightGBM： https://zhuanlan.zhihu.com/p/99069186

读者可以先把参考文章看看

1. **LightGBM和xgBoost、GBDT的区别**☆☆☆☆☆

   参考回答

   xgboost与LightGBM的区别

   1. 切分算法（切分点的选取）：**XGBoost**通过对所有特征都按照特征的数值进行**预排序**选取最好的分割点；**LightGBM**通过**直方图算法**寻找最优的分割点
   2. **LightGBM**占用的内存更低，只保存特征离散化后的值，而这个值一般用8位整型存储就足够了，内存消耗可以降低为原来的1/8
   3. **LightGBM**直接支持类别特征
   4. 决策树生长策略不同

      XGBoost采用的是带深度限制的**level-wise**生长策略。level-wise过一次数据可以能够同时分裂同一层的叶子，容易进行多线程优化，不容易过拟合；但不加区分的对待同一层叶子，带来了很多没必要的开销（实际上很多叶子的分裂增益较低，没必要进行搜索和分裂）

      LightGBM采用**leaf-wise**生长策略，每次从当前所有叶子中找到分裂增益最大（数据量最大）的一个叶子，进行分裂，如此循环；但会生长出比较深的决策树，产生过拟合（因此LightGBM在leaf-wise之上增加了一个最大深度的限制，在保证高效率的同时防止过拟合）

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）。

2. 传统**GBDT**在优化时只用到一阶导数信息，**xgboost**则对代价函数进行了二阶泰勒展开，同时用到了一阶和二阶导数，训练速度更快。
3. **xgboost**在代价函数里加入了正则项，用于控制模型的复杂度。正则项里包含了树的叶子节点个数、每个叶子节点上输出的score的L2模的平方和。防止过拟合
4. Shrinkage（缩减），相当于学习速率（xgboost中的eta）。**xgboost**在进行完一次迭代后，会将叶子节点的权重乘上该系数，主要是为了削弱每棵树的影响，让后面有更大的学习空间。
5. 列抽样（column subsampling）。**xgboost**借鉴了随机森林的做法，支持列抽样，不仅能降低过拟合，还能减少计算，这也是xgboost异于**传统gbdt**的一个特性。对缺失值的处理。对于特征的值有缺失的样本，**xgboost**可以自动学习出它的分裂方向；**传统的GBDT**没有设计对缺失值进行处理
6. **xgboost**工具支持并行。**GBDT**属于串行。
7. **GBDT**是机器学习算法，**XGBoost**是该算法的工程实现。

答案解析

无。

类似的问题还有：

2. **xgb和gbdt的区别**☆☆☆☆☆

   参考回答

   xgBoost、GBDT的区别

   1. 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）。
   2. 传统**GBDT**在优化时只用到一阶导数信息，**xgboost**则对代价函数进行了二阶泰勒展开，同时用到了一阶和二阶导数，训练速度更快。
   3. **xgboost**在代价函数里加入了正则项，用于控制模型的复杂度。正则项里包含了树的叶子节点个数、每个叶子节点上输出的score的L2模的平方和。防止过拟合
   4. Shrinkage（缩减），相当于学习速率（xgboost中的eta）。**xgboost**在进行完一次迭代后，会将叶子节点的权重乘上该系数，主要是为了削弱每棵树的影响，让后面有更大的学习空间。
   5. 列抽样（column subsampling）。**xgboost**借鉴了随机森林的做法，支持列抽样，不仅能降低过拟合，还能减少计算，这也是xgboost异于**传统gbdt**的一个特性。对缺失值的处理。对于特征的值有缺失的样本，**xgboost**可以自动学习出它的分裂方向；**传统的GBDT**没有设计对缺失值进行处理

蒋豆芽

答案解析

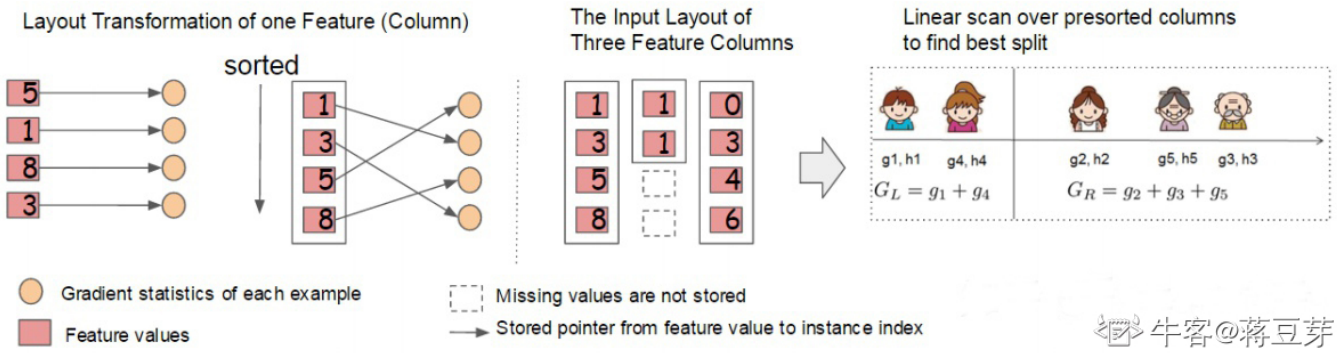
无。

3. **xgBoost的block结构** ☆ ☆ ☆ ☆ ☆

参考回答

在**树**生成过程中，最耗时的一个步骤就是在**每次寻找最佳分裂点时都需要对特征的值进行排序。**而 **XGBoost** 在训练之前会根据特征对数据进行排序，然后保存到**块结构**中，并在每个块结构中都采用了**稀疏矩阵存储格式（Compressed Sparse Columns Format，CSC）**进行存储，后面的训练过程中会重复地使用块结构，可以大大减小计算量。

作者提出通过**按特征进行分块并排序**，在块里面保存排序后的特征值及对应样本的引用，以便于获取样本的一阶、二阶导数值。具体方式如图：



通过顺序访问**排序后的块**遍历样本特征的特征值，方便进行切分点的查找。此外分块存储后多个特征之间互不干涉，可以使用多线程同时对不同的特征进行切分点查找，即特征的**并行化处理。**在对节点进行分裂时需要选择增益最大的特征作为分裂，这时各个特征的增益计算可以同时进行，**这也是 XGBoost 能够实现分布式或者多线程计算的原因。**

答案解析

无。

4. **XGBoost的优缺点** ☆ ☆ ☆ ☆ ☆

参考回答

优点

1. **精度更高**：GBDT 只用到一阶泰勒展开，而 **XGBoost** 对损失函数进行了二阶泰勒展开。XGBoost 引入二阶导一方面是为了增加精度，另一方面也是为了能够自定义损失函数，二阶泰勒展开可以近似大量损失函数；

性回归（回归问题）。此外，XGBoost 工具支持自定义损失函数，只需函数支持一阶和二阶求导；

3. **正则化**：**XGBoost** 在目标函数中加入了正则项，用于控制模型的复杂度。正则项里包含了树的叶子节点个数、叶子节点权重的 L2 范式。正则项降低了模型的方差，使学习出来的模型更加简单，有助于防止过拟合，这也是XGBoost优于传统GBDT的一个特性。
4. **Shrinkage（缩减）**：相当于学习速率。**XGBoost** 在进行完一次迭代后，会将叶子节点的权重乘上该系数，主要是为了削弱每棵树的影响，让后面有更大的学习空间。传统GBDT的实现也有学习速率；
5. **列抽样**：**XGBoost** 借鉴了随机森林的做法，支持列抽样，不仅能降低过拟合，还能减少计算。这也是XGBoost异于传统GBDT的一个特性；
6. **缺失值处理**：对于特征的值有缺失的样本，**XGBoost** 采用的稀疏感知算法可以自动学习出它的分裂方向；
7. **XGBoost工具支持并行**：XGBoost在训练之前，预先对数据进行了排序，然后保存为block结构，后面的迭代中重复地使用这个结构，大大减小计算量。这个block结构也使得并行成为了可能，在进行节点的分裂时，需要计算每个特征的增益，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进行。
8. 可并行的近似算法：树节点在进行分裂时，我们需要计算每个特征的每个分割点对应的增益，即用贪心法枚举所有可能的分割点。当数据无法一次载入内存或者在分布式情况下，贪心算法效率就会变得很低，所以XGBoost还提出了一种可并行的近似算法，用于高效地生成候选的分割点。

缺点

1. 虽然利用预排序和近似算法可以降低寻找最佳分裂点的计算量，**但在节点分裂过程中仍需要遍历数据集**；
2. 预排序过程的空间复杂度过高，不仅需要存储特征值，还需要存储特征对应样本的梯度统计值的索引，相当于消耗了两倍的内存。

答案解析

无。

5. **集成学习 Bagging Boosting**☆☆☆☆☆

参考回答

Bagging 是 bootstrap aggregation的缩写。bagging对于数据集进行**bootstrap取样**，每个数据点有同等几率被采样，然后创建n个模型，每个模型进行m个数据采样，最后进行投票（voting）得出最后结果。**Bagging 的典型应用是随机森林**

蒋豆芽

一些弱学习器串起来，组成一个强学习器。**boosting的典型应用是Adaboost。**

答案解析

无。

6. **RF和GBDT的区别** ☆ ☆ ☆ ☆ ☆

参考回答

RF与GBDT之间的区别

（1）**相同点**

1. 都是由多棵树组成
2. 最终的结果都是由多棵树一起决定

（2）**不同点**

1. 组成随机森林的树可以分类树也可以是回归树，而GBDT只由回归树组成
2. 组成随机森林的树可以并行生成，而GBDT是串行生成
3. 随机森林的结果是多数表决表决的，而GBDT则是多棵树累加之和
4. 随机森林对异常值不敏感，而GBDT对异常值比较敏感
5. 随机森林是通过减少模型的方差来提高性能，而GBDT是减少模型的偏差来提高性能的
6. 随机森林不需要进行数据预处理，即特征归一化。而GBDT则需要进行特征归一化

答案解析

无。

7. **GBDT是否适合于处理大规模的ID特征** ☆ ☆ ☆ ☆ ☆

参考回答

GBDT对于海量的id类特征，GBDT由于树的深度和树的数量限制（防止过拟合），不能有效存储；另外海量特征也会存在性能瓶颈，当GBDT的one hot特征大于100k维时，需要做分布式训练才能保证不爆内存，因此，GBDT通常配合少量的反馈CTR特征来表达，在带来一定范化能力的同时会有信息损失，对于头部资源无法有效表达。

答案解析

无。

参考回答

1. 基于树模型的boosting算法，很多算法比如xgboost都是用**预排序（pre-sorting）算法**进行特征的选择和分裂
2. LightGBM采用**HistoGram算法**，其思想是将连续的浮点特征离散成k个离散值，并构造宽度为k的Histogram。然后遍历训练数据，计算每个离散值在直方图中的累计统计量。在进行特征选择时，只需要根据直方图的离散值，遍历寻找最优的分割点。

所以如果LightGBM的直方图排序后，最优的分割点就变了，效果可能会比xgboost差。

答案解析

无。

9. **xgboost正则化项和什么有关** ☆ ☆ ☆ ☆ ☆

参考回答

正则项里包含了树的叶子节点个数、每个叶子节点上输出的score的L2模的平方和。

答案解析

无。

10. **随机森林哪两个随机** ☆ ☆ ☆ ☆ ☆

参考回答

2个随机(bootstrap+特征m)

1. 应用 bootstrap 法有放回地随机抽取 k个新的自助样本集(boostrap)，并由此构建 k 棵分类树(ID3 、 C4.5 、 CART)样本扰动。
2. 先随机选择属性子集，个数为k，然后再从这个子集中选择一个最优属性用于划分。

答案解析

无。

11. **bootstrap怎么做的** ☆ ☆ ☆ ☆ ☆

参考回答

答案解析

无。

12. **介绍GBDT的详细计算过程** ☆ ☆ ☆ ☆ ☆

参考回答

回答参考文章。

答案解析

无。

13. **xgb的正则项是什么** ☆ ☆ ☆ ☆ ☆

参考回答

正则项里包含了树的叶子节点个数、每个叶子节点上输出的score的L2模的平方和。

答案解析

无。

14. **xgboost缺失值处理方法** ☆ ☆ ☆ ☆ ☆

参考回答

论文中关于**缺失值的处理**将其看与**稀疏矩阵**的处理看作一样。在寻找split point的时候，不会对该特征为missing的样本进行遍历统计，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，会分别处理将missing该特征值的样本分配到左叶子结点和右叶子结点的两种情形，计算增益后选择增益大的方向进行分裂即可。可以为缺失值或者指定的值指定分支的默认方向，这能大大提升算法的效率。**如果在训练中没有缺失值而在预测中出现缺失，那么会自动将缺失值的划分方向放到右子树。**

答案解析

无。

15. **为什么xgboost要二阶展开?** ☆ ☆ ☆ ☆ ☆

+二阶项的形式，而其他类似log loss这样的目标函数不能表示成这种形式。为了后续推导的统一，所以将目标函数进行二阶泰勒展开，就可以直接自定义损失函数了，只要二阶可导即可，增强了模型的扩展性。

2. **二阶信息能够让梯度收敛的更快**，类似牛顿法比SGD收敛更快。一阶信息描述梯度变化方向，二阶信息可以描述梯度变化方向是如何变化的。

答案解析

无。

16. **集成学习的方法有哪些**☆☆☆☆☆

参考回答

回答参考上面。

答案解析

无。

17. **泰勒公式求e的近似值**☆☆☆☆☆

参考回答

自然常数 e 可以用级数 1+1/1!+1/2!+…+1/n!+… 来近似计算。

答案解析

无。

18. **XGBoost 如果损失函数没有二阶导，该怎么办**☆☆☆☆☆

参考回答

gbdt的目标函数与**xgboost**区别就是带不带**正则项**(算法内容上)。**gbdt**对损失函数的优化是直接使用了损失函数的负梯度，沿着梯度下降的方向来减小损失，其是也就是一阶泰勒展开。而**xgboost**在这里使用了**二阶泰勒展开**，因为包含了损失函数的二阶信息，其优化的速度大大加快。**但如果loss没有二阶导数，就使用一阶导数优化**

答案解析

无。

蒋豆芽

参考回答

样本数

答案解析

无。

资源分享 python 机器学习 算法工程师 春秋招 面试题 软件开发 面经 举报

收藏 赞

相关专栏

机器学习面试题汇总与解析（蒋豆芽面试题总结）

27篇文章 | 90订阅

已订阅

1条评论 默认排序

蒋豆芽 N

2021-07-10 11:59:24

0 0 1#

550681844，大家可以加入这个群，里面有学习资料下载，希望能帮到大家。问题答案：牛客网

请留下你的观点吧~

发布

蒋豆芽