题库 | 为加薪而战！随机森林、GBDT、代价函数等面试题你都掌握了？

原创 专注AI求职服务的 AI职通车 2019-10-17

记得点击蓝字，关注我们呀~

每天十道机器学习专项面试题介绍，助你轻轻松松收获offer~

-机器学习-

随机森林算法是如何工作的？

在随机森林中，每一个决策树“种植”和“生长”的规则如下所示：

1. 假设我们设定训练集中的样本个数为 N，然后通过有重置的重复多次抽样来获得这 N 个样本，这样的抽样结果将作为我们生成决策树的训练集；

2. 如果有 M 个输入变量，每个节点都将随机选择 m (m<M) 个特定的变量，然后运用这 m 个变量来确定最佳的分裂点。在决策树的生成过程中，m 的值是保持不变的；

3. 每棵决策树都最大可能地进行生长而不进行剪枝；

4. 通过对所有的决策树进行加总来预测新的数据（在分类时采用多数投票，在回归时采用平均）。

随机森林分类效果的影响因素是什么？

森林中任意两棵树的相关性：相关性越大，错误率越大；

森林中每棵树的分类能力：每棵树的分类能力越强，整个森林的错误率越低。

减小特征选择个数m，树的相关性和分类能力也会相应的降低；增大m，两者也会随之增大。所以关键问题是如何选择最优的m（或者是范围），这也是随机森林唯一的一个参数。

随机森林有什么优缺点?

优点：

- 在当前的很多数据集上，相对其他算法有着很大的优势，表现良好。
- 它能够处理很高维度（feature很多）的数据，并且不用做特征选择(因为特征子集是随机选择的)。
- 在训练完后，它能够给出哪些feature比较重要。
- 训练速度快，容易做成并行化方法(训练时树与树之间是相互独立的)。
- 在训练过程中，能够检测到feature间的互相影响。
- 对于不平衡的数据集来说，它可以平衡误差。
- 如果有很大一部分的特征遗失，仍可以维持准确度。

缺点：

- 随机森林在解决回归问题时并没有像它在分类中表现的那么好，这是因为它并不能给出一个连续型的输出。当进行回归时，随机森林不能够作出超越训练集数据范围的预测，这可能导致在对某些还有特定噪声的数据进行建模时出现过度拟合。
- 对于许多统计建模者来说，随机森林给人的感觉像是一个黑盒子 —— 你几乎无法控制模型内部的运行，只能在不同的参数和随机种子之间进行尝试。

随机森林如何处理缺失值?

根据随机森林创建和训练的特点，随机森林对缺失值的处理还是比较特殊的。

- 首先，给缺失值预设一些估计值，比如数值型特征，选择其余数据的中位数或众数作为当前的估计值
- 然后，根据估计的数值，建立随机森林，把所有的数据放进随机森林里面跑一遍。记录每一组数据在决策树中一步一步分类的路径.

- 判断哪组数据和缺失数据路径最相似，引入一个相似度矩阵，来记录数据之间的相似度，比如有N组数据，相似度矩阵大小就是N*N
- 如果缺失值是类别变量，通过权重投票得到新估计值，如果是数值型变量，通过加权平均得到新的估计值，如此迭代，直到得到稳定的估计值。

其实，该缺失值填补过程类似于推荐系统中采用协同过滤进行评分预测，先计算缺失特征与其他特征的相似度，再加权得到缺失值的估计，而随机森林中计算相似度的方法（数据在决策树中一步一步分类的路径）乃其独特之处。

什么是OOB？随机森林中OOB是如何计算的，它有什么优缺点？

OOB：

上面我们提到，构建随机森林的关键问题就是如何选择最优的m，要解决这个问题主要依据计算袋外错误率oob error（out-of-bag error）。

bagging方法中Bootstrap每次约有1/3的样本不会出现在Bootstrap所采集的样本集合中，当然也就没有参加决策树的建立，把这1/3的数据称为袋外数据oob（out of bag）,它可以用于取代测试集误差估计方法。

袋外数据(oob)误差的计算方法如下：

- 对于已经生成的随机森林,用袋外数据测试其性能,假设袋外数据总数为O,用这O个袋外数据作为输入,带进之前已经生成的随机森林分类器,分类器会给出O个数据相应的分类
- 因为这O条数据的类型是已知的,则用正确的分类与随机森林分类器的结果进行比较,统计随机森林分类器分类错误的数目,设为X,则袋外数据误差大小=X/O

优缺点：

这已经经过证明是无偏估计的,所以在随机森林算法中不需要再进行交叉验证或者单独的测试集来获取测试集误差的无偏估计。

RF(随机森林)与GBDT之间的区别与联系？

相同点：

- 都是由多棵树组成，最终的结果都是由多棵树一起决定。

- RF和GBDT在使用CART树时，可以是分类树或者回归树。

不同点：

- 组成随机森林的树可以并行生成，而GBDT是串行生成
- 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
- 随机森林对异常值不敏感，而GBDT对异常值比较敏感
- 随机森林是减少模型的方差，而GBDT是减少模型的偏差
- 随机森林不需要进行特征归一化。而GBDT则需要进行特征归一化

解释一下GBDT算法的过程？

GBDT(Gradient Boosting Decision Tree)，全名叫梯度提升决策树，使用的是Boosting的思想。

Boosting 的思想：

Boosting 方法训练基分类器时采用串行的方式，各个基分类器之间有依赖。它的基本思路是将基分类器层层叠加，每一层在训练的时候，对前一层基分类器分错的样本，给予更高的权重。测试时，根据各层分类器的结果的加权得到最终结果。

Bagging 与 Boosting 的串行训练方式不同，Bagging 方法在训练过程中，各基分类器之间无强依赖，可以进行并行训练。

GBDT 的原理很简单，就是所有弱分类器的结果相加等于预测值，然后下一个弱分类器去拟合误差函数对预测值的残差(这个残差就是预测值与真实值之间的误差)。当然了，它里面的弱分类器的表现形式就是各棵树。

举一个非常简单的例子，比如我今年30岁了，但计算机或者模型 GBDT 并不知道我今年多少岁，那 GBDT 咋办呢？

- 它会在第一个弱分类器（或第一棵树中）随便用一个年龄比如20岁来拟合，然后发现误差有10岁；
- 接下来在第二棵树中，用6岁去拟合剩下的损失，发现差距还有4岁；
- 接着在第三棵树中用3岁拟合剩下的差距，发现差距只有1岁了；
- 最后在第四课树中用1岁拟合剩下的残差，完美。

- 最终，四棵树的结论加起来，就是真实年龄30岁（实际工程中，gbdt是计算负梯度，用负梯度近似残差）。

为何 GBDT 可以用负梯度近似残差呢?

回归任务下，GBDT 在每一轮的迭代时对每个样本都会有一个预测值，此时的损失函数为均方差损失函数，

$$l(y_i, y^i) = \frac{1}{2}(y_i - y^i)^2$$

那此时的负梯度是这样计算的

$$-[\frac{\partial l(y_i, y^i)}{\partial y^i}] = (y_i - y^i)$$

所以，当损失函数选用均方损失函数是时，每一次拟合的值就是（真实值 - 当前模型预测的值），即残差。此时的变量是y^i，即"当前预测模型的值"，也就是对它求负梯度。

GBDT的优点和局限性有哪些?

优点：

- 预测阶段的计算速度快，树与树之间可并行化计算。
- 在分布稠密的数据集上，泛化能力和表达能力都很好，这使得GBDT在Kaggle的众多竞赛中，经常名列榜首。
- 采用决策树作为弱分类器使得GBDT模型具有较好的解释性和鲁棒性，能够自动发现特征间的高阶关系。

局限性：

- GBDT在高维稀疏的数据集上，表现不如支持向量机或者神经网络。
- GBDT在处理文本分类特征问题上，相对其他模型的优势不如它在处理数值特征时明显。
- 训练过程需要串行训练，只能在决策树内部采用一些局部并行的手段提高训练速度。

为什么需要代价函数？

1. 为了得到训练逻辑回归模型的参数，需要一个代价函数，通过训练代价函数来得到参数。
2. 用于找到最优解的目的函数。

为什么代价函数要非负？

目标函数存在一个下界，在优化过程当中，如果优化算法能够使目标函数不断减小，根据单调有界准则，这个优化算法就能证明是收敛有效的。只要设计的目标函数有下界，基本上都可以，代价函数非负更为方便。

常见代价函数有哪些？

二次代价函数（quadratic cost）：

$$J = \frac{1}{2n}\sum_x \|y(x) - a^L(x)\|^2$$

其中，J 表示代价函数，x 表示样本，y 表示实际值，a 表示输出值，n 表示样本的总数。使用一个样本为例简单说明，此时二次代价函数为：

$$J = \frac{(y-a)^2}{2}$$

假如使用梯度下降法（Gradient descent）来调整权值参数的大小，权值 w 和偏置 b 的梯度推导如下：

$$\frac{\partial J}{\partial b} = (a-y)\sigma'(z)$$

其中，z 表示神经元的输入，σ 表示激活函数。权值 w 和偏置 b 的梯度跟激活函数的梯度成正比，激活函数的梯度越大，权值 w 和偏置 b 的大小调整得越快，训练收敛得就越快。

注：神经网络常用的激活函数为 sigmoid 函数，该函数的曲线如下图所示：

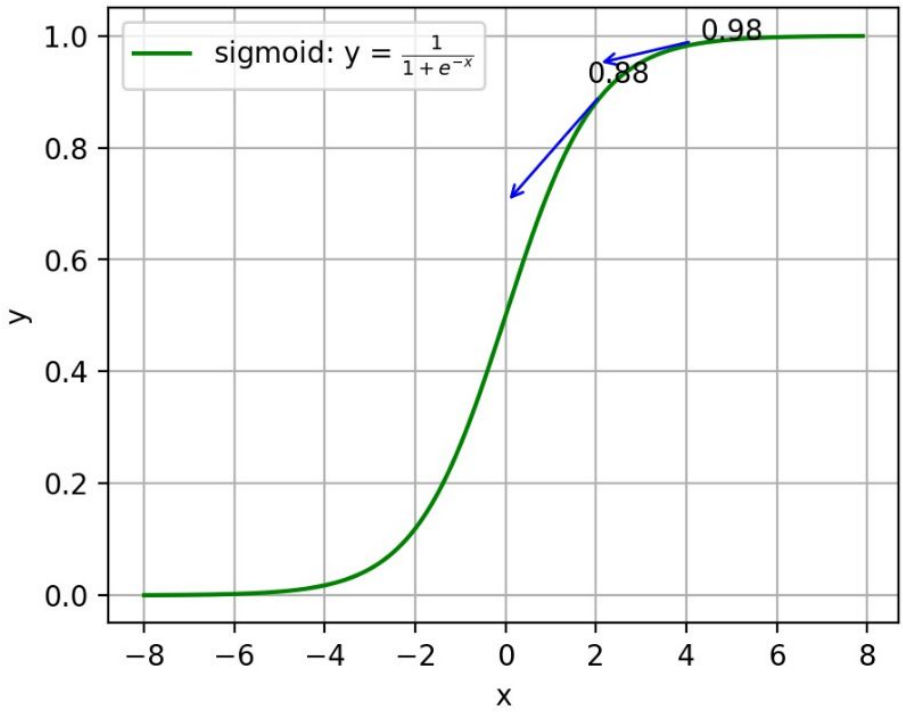

图 sigmoid 函数曲线

如上图所示，对 0.88 和 0.98 两个点进行比较：假设目标是收敛到 1.0。0.88 离目标 1.0 比较远，梯度比较大，权值调整比较大。0.98 离目标 1.0 比较近，梯度比较小，权值调整比较小。调整方案合理。假如目标是收敛到 0。0.88 离目标 0 比较近，梯度比较大，权值调整比较大。0.98 离目标 0 比较远，梯度比较小，权值调整比较小。调整方案不合理。原因：在使用 sigmoid 函数的情况下，初始的代价（误差）越大，导致训练越慢。

交叉熵代价函数（cross-entropy）：

$$J=-\frac{1}{n}\sum_x[y\ln a+(1-y)\ln(1-a)]$$

其中，J 表示代价函数，x 表示样本，y 表示实际值，a 表示输出值，n 表示样本的总数。权值 w 和偏置 b 的梯度推导如下：$$ \frac {\partial J}{\partial w_j}=\frac {1}{n}\sum_{x} x_j (\sigma {(z)}-y);， \frac {\partial J}{\partial b}=\frac {1}{n}\sum_{x}(\sigma {(z)}-y) $$

当误差越大时，梯度就越大，权值 w 和偏置 b 调整就越快，训练的速度也就越快。**二次代价函数适合输出神经元是线性的情况，交叉熵代价函数适合输出神经元是 S 型函数的情况。**

对数似然代价函数（log-likelihood cost）：

对数似然函数常用来作为 softmax 回归的代价函数。深度学习中普遍的做法是将 softmax 作为最后一层，此时常用的代价函数是对数似然代价函数。对数似然代价函数与 softmax 的组合和交叉熵与 sigmoid 函数的组合非常相似。对数似然代价函数在二分类时可以化简为交叉熵代价函数的形式。

在 tensorflow 中：

- 与 sigmoid 搭配使用的交叉熵函数：

  tf.nn.sigmoid_cross_entropy_with_logits()。

- 与 softmax 搭配使用的交叉熵函数：

  tf.nn.softmax_cross_entropy_with_logits()。

在 pytorch 中：

- 与 sigmoid 搭配使用的交叉熵函数：torch.nn.BCEWithLogitsLoss()。
- 与 softmax 搭配使用的交叉熵函数：torch.nn.CrossEntropyLoss()。

- 等你来答 -

1、如果你有一个数百万特征的训练集，你应该选择哪种线性回归训练算法?

2、逻辑回归常用的优化方法有哪些?

3、决策树如何避免过拟合?

4、如何解决线性不可分问题?

5、为如何处理数据中的缺失值?

如果你想要回答问题，请参考下方的认领方式，或通过求职百题斩小程序答题噢～

认领步骤

选择一个你感兴趣的。

然后在评论下方作答。

作答格式：标题 + 问题解析

题库系列会长久更新，直到上千道、甚至数万道题~

希望各位来分享你在求职面试当中遇到的难题、趣题~

如你有私人收藏的珍贵好题，欢迎来勾搭题库小哥哥（微信ID：Huanger07）

互动福利

哈喽大家好~，这里是AI研习社无所不能的社长，不过社长最近有个疑问，想征集一些大家的看法，欢迎大家来和社长互动，帮社长解答一下这个疑惑呀~

社长的疑问是：**作为求职者，你最关心的问题是什么？**

为答谢回答问题的朋友，社长准备每天提供一张价值**100元**的AI慕课学院通用优惠券。

那么，什么是AI慕课学院呢？

AI慕课学院专注于**人工智能人才培训**，汇集行业知名专家，结合产业特色，为高校学生、人工智能爱好者和从业者、科技行业职业人群提供低学习门槛的人工智能培训课程，就机器学习、模式识别、数据挖掘、自然语言处理、计算机视觉、人机交互等人工智能相关主题进行研讨和授课，为学习者提供系统的进阶教育。

参与方式：**关注**本公众号（AI职通车），在本文底部进行**留言**，每天点赞数第一的朋友将获得价值100元的AI慕课学院通用优惠券，希望大家畅所欲言，说出你对这个问题的看法吧~

我们希望天下学AI者都有用武之地！

AI职通车还在成长中，期待你的加入~

点击 **阅读原文**，查看更多 AI 岗位详情。

阅读原文