CSDN 博客 下载 学习 社区 问答 GitCode HelloGitHub Boolean 登录 注册 会员中心 消息 创作 创作中心 发布

Jaime-E 关注
2024-07-25 09:57:00 阅读量:1.2k 收藏量:24 点赞量:24 分享数:24
分类专栏: Deep Learning 文章标签: GBDT Xgboost 原理 Boosting

Deep Learning 专栏收录该内容 订阅 14 订阅 14 篇文章 已收专栏

GBDT和Xgboost：原理、推导、比较

写在前面

GBDT和Xgboost是机器学习中的两个重要算法，它们在数据分析和预测任务中表现出色。GBDT（梯度提升决策树）是一种基于决策树的集成学习方法，通过逐步优化损失函数来提升模型的预测性能。Xgboost（极端梯度提升）是GBDT的一种高效实现，它在保持GBDT核心思想的基础上，进一步优化了计算效率和内存使用，以及处理大规模数据的能力。

在最新版本的工具库中，GBDT和Xgboost的实现方式有所不同。GBDT通常作为一个独立的模块或类来实现，而Xgboost则提供了更多的参数和接口，以支持更灵活的模型调优和特征工程。

提示：GBDT和Xgboost的区别是不可或缺的！

- 三棵树的残差加权求和：每一次迭代的残差都会被加权求和，GBDT和Xgboost不同。
- 正则化项：GBDT没有正则化项，Xgboost则通过引入正则化项来控制模型的复杂度，以防止过拟合。
- 二阶导数：GBDT仅使用一阶导数来计算梯度，而Xgboost则利用了二阶导数来提高计算效率和精度。
- 缺失值处理：GBDT通常需要手动处理缺失值，而Xgboost则能够自动处理缺失值，并为其学习最优的分裂方向。
- 并行计算：GBDT的实现通常是单线程的，而Xgboost则支持多线程并行计算，以加速模型的训练过程。
- 内存优化：Xgboost在内存使用上进行了优化，以减少内存占用，特别是在处理大规模数据集时。
- 特征重要性：GBDT和Xgboost都可以计算特征的重要性，但Xgboost提供了更多的选项和更精确的计算方法。

一、GBDT

1. 思想

GBDT（梯度提升决策树）的核心思想是通过逐步优化损失函数来提升模型的预测性能。它通过以下几个关键步骤来实现：

1. 初始化：从一个简单的模型开始，通常是一个常数模型。
2. 计算残差：计算当前模型的预测值与实际值之间的差异，这些差异被称为残差。
3. 拟合残差：使用一个新的决策树来拟合这些残差，以减少预测误差。
4. 更新模型：将新的决策树加权后加入到模型中，并更新模型的预测值。
5. 重复：重复上述步骤，直到达到预设的迭代次数或模型性能不再显著提升为止。

2. 梯度残差的来源

为什么要拟合残差？这是因为残差代表了模型当前预测与真实值之间的差异，通过拟合残差，模型可以逐步减小这个差异，从而提高预测的准确性。残差的来源主要包括：

- 模型的不足：当前模型可能无法完全捕捉数据中的模式，导致预测结果与真实值存在偏差。
- 噪声：数据中可能存在噪声，这些噪声会导致模型的预测结果与真实值不一致。
- 特征的影响：某些特征可能对目标变量有重要影响，但当前模型可能未能充分利用这些特征。

3. 梯度残差的计算

在GBDT中，残差通常是指负梯度。对于一个损失函数 $L(y, F(x))$，其中 y 是真实值，$F(x)$ 是模型的预测值，残差可以表示为：

$$r = -\frac{\partial L(y, F(x))}{\partial F(x)}$$

这个公式表示残差是损失函数对模型预测值的负梯度。通过这个方向，模型可以调整预测值，以减小损失函数的值。

4. 梯度提升的原理

梯度提升（Gradient Boosting）是一种通过逐步优化损失函数来提升模型性能的方法。其核心思想是通过拟合残差来逐步减小预测误差。具体来说，梯度提升通过以下步骤来实现：

1. 初始化：从一个简单的模型开始，通常是一个常数模型。
2. 计算残差：计算当前模型的预测值与实际值之间的差异，这些差异被称为残差。
3. 拟合残差：使用一个新的决策树来拟合这些残差，以减少预测误差。
4. 更新模型：将新的决策树加权后加入到模型中，并更新模型的预测值。
5. 重复：重复上述步骤，直到达到预设的迭代次数或模型性能不再显著提升为止。

梯度提升的核心在于通过逐步优化损失函数来提升模型的预测性能。每一步都通过拟合残差来减小预测误差，从而逐步提高模型的准确性。这种方法不仅适用于回归问题，也适用于分类问题，通过调整损失函数和模型结构，可以处理各种复杂的数据模式。

5. 梯度提升决策树

梯度提升决策树（Gradient Boosting Decision Tree，简称GBDT）是一种结合了决策树和梯度提升方法的机器学习算法。它通过逐步优化损失函数来提升模型的预测性能，特别是在处理复杂数据模式时表现出色。GBDT的核心思想是通过拟合残差来逐步减小预测误差，从而提高模型的准确性。

6. 梯度提升决策树的优点

- 高精度：GBDT通过逐步优化损失函数，能够有效地减少预测误差，从而提高模型的准确性。
- 处理非线性关系：决策树本身能够捕捉非线性关系，GBDT通过组合多棵决策树，进一步增强了模型对非线性关系的处理能力。
- 特征选择：GBDT在构建决策树的过程中，会自动选择重要特征，这有助于减少模型的复杂度，并提高模型的可解释性。
- 鲁棒性：GBDT对噪声和异常值具有一定的鲁棒性，因为它通过拟合残差来逐步减小误差，而单个噪声或异常值的影响会被其他决策树所抵消。

7. 梯度提升决策树的缺点

- 计算复杂度：GBDT需要逐步构建多棵决策树，每棵决策树都需要计算残差并更新模型，这可能导致计算时间较长，特别是在处理大规模数据集时。
- 内存消耗：由于需要存储多棵决策树的结构和参数，GBDT可能会占用较多内存，特别是在处理高维数据时。
- 参数调优：GBDT的性能对参数设置非常敏感，需要通过交叉验证等方法来调优参数，以避免过拟合或欠拟合。

8. 梯度提升决策树的应用场景

- 回归问题：GBDT在回归问题中表现出色，能够有效地减少预测误差，从而提高模型的准确性。
- 分类问题：GBDT也可以用于分类问题，通过调整损失函数和模型结构，可以处理各种复杂的分类任务。
- 特征工程：GBDT在特征选择和特征重要性评估方面表现优秀，能够帮助数据科学家发现关键特征，从而优化模型的性能。

二、Xgboost

Xgboost是GBDT的一种高效实现，它在保持GBDT核心思想的基础上，进一步优化了计算效率和内存使用，以支持大规模数据集的处理。Xgboost通过引入正则化项来控制模型的复杂度，以防止过拟合，并通过多线程并行计算来加速模型的训练过程。此外，Xgboost还提供了更多的参数和接口，以支持更灵活的模型调优和特征工程。

1. 目标函数

对于一般的模型，其目标函数定义为：

$$Obj = \sum_{i=1}^{n} L(y_i, F(x_i)) + \Omega(F(x))$$

对于提升树模型，$F(x_i) = F_{t-1}(x_i) + f_t(x_i)$，其中 $f_t(x_i)$ 是当前加入的新模型，此时，目标函数可以写成：

$$Obj^{(t)} = \sum_{i=1}^{n} L(y_i, F_{t-1}(x_i) + f_t(x_i)) + \Omega(f_t) + \text{常数}$$

此时最优化目标函数 $Obj^{(t)}$，即最优化当前的新模型 $f_t(x_i)$，这也说明了Boosting的思想是基本从某新模型子目标逐步拟合残差的样本。

2. 引入泰勒公式

将 $Obj^{(t)}$ 在 $F_{t-1}(x_i)$ 处的泰勒公式展开：

$$Obj^{(t)} \approx \sum_{i=1}^{n} L(y_i, F_{t-1}(x_i)) + L'(y_i, F_{t-1}(x_i))f_t(x_i) + \frac{1}{2}L''(y_i, F_{t-1}(x_i))f_t(x_i)^2 + \Omega(f_t(x_i)) + \text{常数}$$

则可以令 $L'(y_i, F_{t-1}(x_i)), h_i = \frac{1}{2}L''(y_i, F_{t-1}(x_i))$，因为常数不影响最优值，所以为了简单，我们将常数项去掉：
$L(y_i, F_{t-1}(x_i))$ 是一个常数，所以我们可以直接去掉，则：

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

由于 $F_{t-1}(x_i)$ 是一次迭代产生的结果，g_i 已知，所以最优化 $Obj^{(t)}$ 就等价于最优化 $f_t(x_i)$。

3. 正则化表达式

假设决策树有 T 片叶子，每片叶子对应的值分别是 w^T，在决策树中，每一个值的预测结果都是一样的，那么一定存在一个函数：$R^T \to \{1, 2, ..., T\}$，其可以将已经的每个样本数据分配到一个叶子结点上，那么正式的定义可以写成：$q(x_i)$ 表示 x_i 的叶子结点，$w_{q(x_i)}$ 则代表了该叶子结点的权重，所以 $w_{q(x_i)}$ 就代表了每个样本的最终值（预测值）。

决策树复杂度可以定义为：

$$\Omega(f_t(x_i)) = \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2$$

即决策树复杂度由叶子结点数量和叶子节点数量的平方和共同决定。

4. 正则化表达式

假设决策树有 T 片叶子，每片叶子对应的值分别是 w^T，在决策树中，每一个值的预测结果都一样，那么一定存在一个函数：$R^T \to \{1, 2, ..., T\}$，其可以将已经的每个样本数据分配到一个叶子结点上，那么正式的定义可以写成：$q(x_i)$ 表示 x_i 的叶子结点，$w_{q(x_i)}$ 则代表了该叶子结点的权重，所以 $w_{q(x_i)}$ 就代表了每个样本的最终值（预测值）。

$$\Omega(f_t(x_i)) = \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2$$

即决策树复杂度由叶子结点数量和叶子节点数量的平方和共同决定。

5. 正则化表达式

将 $\Omega(f_t(x_i))$ 代入 $Obj^{(t)}$，则：

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

$$= \sum_{i=1}^{n} g_i f_t(x_i) + h_i f_t(x_i)^2 + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2$$

$$= \sum_{j=1}^{T}[g_i w_j + \sum_{i \in I_j} h_i w_j^2] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2$$

$$= \sum_{j=1}^{T}[g_i w_j + (\sum_{i \in I_j} h_i + \lambda)w_j^2] + \gamma T$$

令 $G_j = \sum_{i \in I_j} g_i$，$H_j = \sum_{i \in I_j} h_i$，则上式可以写成：

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

如果我们已经确定了树的结构，即必要确定的，对应着我们已经知道了每个叶子结点有哪些样本，则 G_j 和 H_j 是确定的，但是 w 不确定，那么对 w_j 求导令其为零：

$$G_j + (H_j + \lambda)w_j = 0$$
$$\Rightarrow w_j^* = -\frac{G_j}{H_j + \lambda}$$

将此 w_j^* 代入 $Obj^{(t)}$，则：

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

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

计算贪婪的GBDT代表了当前这一个树的结构的时候，目标函数的最基本可以减少多少，也就是增益的含义，一般称其为结构函数。

6. 目标函数求解

对于单棵决策树，一种暴露的状况是既要检测有可能的结构结构，然后用每一棵最优树，过程如下：

Step1 枚举所有可能的结构树，即：
Step2 计算每种结构树下的目标函数，即 $Obj^{(t)}$的值，
Step3 取目标函数值的最小值，使用 w_j^* 求解每个叶子节点的权重，即树中的预测值

但是因为结构的可能性，即枚举所有结构树的结构是实际不可行的，所以一般使用贪心策略来求解：每次尝试分出一个叶节点，计算分裂后的增益，选择最优增益，具体过程如下：

Step1 从深度为0的树开始，对每个叶子节点枚举所有可用特征，
Step2 对每个特征，按照不同取值大小排序，从小到大扫描，计算分裂后的增益，
Step3 选择增益最大的特征作为分裂特征，同时特征取值分裂点作为分裂阈值，更新节点信息及结构下新的叶子节点，对为每个新叶子节点递归执行步骤1-3，
Step4 当满足预设的停止条件（如树深度达到最大值）时，停止生长，

计算增益的公式可以写成：

$$Gain = \frac{1}{2}[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}] - \gamma$$

Gain是计算分裂的增益，这个公式通过比较左和右子节点的梯度和来评估分裂的效果。GAIN越大表示分裂带来的增益越大，因此在实际应用中，通常会选择Gain最大的分裂方案。Gain的计算涉及多个参数，其中，λ 和 γ 是正则化参数，用于控制模型的复杂度，以防止过拟合。

为了简化计算，引入了一个新的变量 η，用来调整学习速率，以控制每一步更新的幅度。这样，目标函数的计算公式可以表示为：

$$Gain = \frac{1}{2}[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}] - \gamma$$

这个公式通过调整 η 的值，可以在模型训练过程中动态调整学习速率，从而提高模型的收敛速度和稳定性。

三、梯度提升：GBDT和Xgboost区别

Random Forest和GBDT区别如下：

- 三种树都是多棵可以最终表现为可以容忍过拟合，GBDT只能容忍过拟合，
- GBDT不同数据分布能力强，GBDT分布能力，
- 三种树都是多棵，三种树都是多棵，加权投票等，而GBDT是将所有结果（加权）累加起来，
- GBDT可以容易不稳定，GBDT可容易稳定，
- 多种可以随机一组特征，GBDT是不Bootstrap抽样，基于残差，分类器越好，权值越小
- 多种是集成决策方案，所以结果呈现方式是三种集成方法，而GBDT主要呈现结果方式，
- 三种随机性结果，GBDT使用结果呈现方式

Xgboost是GBDT的一种，所以Xgboost和GBDT的区别和GBDT一样。

GBDT和Xgboost的区别如下：

- **三棵分类器的选择：**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正则化项的逻辑回归（分类问题）或线性回归（回归问题）。
- **目标函数：**GBDT是一阶导数信息，Xgboost对目标函数进行了泰勒展开，同时用到了一阶和二阶导数信息，两对目标函数进行了二次近似，求得更精确的解。
- **正则项：**GBDT没有正则项，Xgboost在目标函数中加入了正则项，控制模型的复杂度。Gain的计算中，加入了 λ 和 γ 两个正则项，分别对叶子节点的权重和叶子节点数进行惩罚，防止过拟合。
- **正则参数：**Xgboost的正则参数，λ 和 γ 分别对叶子节点的权重和叶子节点数进行惩罚，防止过拟合。
- **缺失值处理：**GBDT需要手动处理缺失值，而Xgboost能够自动处理缺失值，并为其学习最优的分裂方向。
- **并行计算：**GBDT的实现通常是单线程的，而Xgboost则支持多线程并行计算，以加速模型的训练过程。
- **内存优化：**Xgboost在内存使用上进行了优化，以减少内存占用，特别是在处理大规模数据集时。
- **特征重要性：**Xgboost提供了更多的选项和更精确的计算方法，以支持更灵活的模型调优和特征工程。

参考文献：

[1] A Kaggle Master Explains Gradient Boosting
[2] GBDT (Gradient Boosting Decision Tree)
[3] 机器学习——之梯度GBDT的理解 2017-05-01
[4] GBDT算法原理深入解析
[5] GBDT算法原理深入浅出分析合集
[6] 机器学习算法系列（31）：XgBoost
[7] pandas怎么为什么用分类数据？
[8] Friedman J H. Greedy function approximation: a gradient boosting machine[J]. Annals of statistics, 2001: 1189-1232.
[9] Chen T, Guestrin C. Xgboost: A scalable tree boosting system[C]//Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. ACM, 2016: 785-794.

12_AdaBoost_GBDT_Xgboost决策树算法笔记 08-03
1. 为了保证模型的泛化能力，不能过分地追求训练数据的拟合，使得训练数据出现多分类器的评分不稳定 2. 为基本分类器的结合线性组合的权重的分量要小，这样可以使得最终的分类器的评分稳定 3. 使用...
GBDT_XGBoost_LGBM算法笔记 04-03
GBDT决策树与其衍生算法的一些理解记录 参数记录 参数调优方法 决策树算法的一般流程 决策树算法的一般流程 决策树算法的一般流程