XGBoost算法原理小结

原创 石头 机器学习算法那些事 2018-12-24

前言

XGBoost（eXtreme Gradient Boosting）全名叫极端梯度提升，XGBoost是集成学习方法的王牌，在Kaggle数据挖掘比赛中，大部分获胜者用了XGBoost，XGBoost在绝大多数的回归和分类问题上表现的十分顶尖，本文较详细的介绍了XGBoost的算法原理。

目录

最优模型的构建方法

构建最优模型的一般方法是最小化训练数据的损失函数，我们用字母 L表示，如下式：

$$\min_{f \in F} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) \qquad (1)$$

其中，F是假设空间

式（1）称为经验风险最小化，训练得到的模型复杂度较高。当训练数据较小时，模型很容易出现过拟合问题。

因此，为了降低模型的复杂度，常采用下式：

$$\min_{f \in F} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda J(f) \qquad (2)$$

其中J(f)为模型的复杂度，式（2）称为结构风险最小化，结构风险最小化的模型往往对训练数据以及未知的测试数据都有较好的预测 。

应用：决策树的生成和剪枝分别对应了经验风险最小化和结构风险最小化，XGBoost的决策树生成是结构风险最小化的结果，后续会详细介绍。

Boosting方法的回归思想

Boosting法是结合多个弱学习器给出最终的学习结果，不管任务是分类或回归，我们都用回归任务的思想来构建最优Boosting模型 。

回归思想：把每个弱学习器的输出结果当成连续值，这样做的目的是可以对每个弱学习器的结果进行累加处理，且能更好的利用损失函数来优化模型。

假设 $f^t(x_i)$ 是第 t 轮弱学习器的输出结果，$\hat{y}_i^t$ 是模型的输出结果，y_i 是实际输出结果，表达式如下：

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f^k(x_i) \qquad (2.1)$$

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f^t(x_i) \qquad (2.2)$$

上面两式就是加法模型，都默认弱学习器的输出结果是连续值。因为回归任务的弱学习器本身是连续值，所以不做讨论，下面详细介绍分类任务的回归思想。

分类任务的回归思想：

根据2.1式的结果，得到最终的分类器：

$$G(x) = sign(\hat{y}_i^{(t)}) = sign(\sum_{k=1}^{t} f^k(x_i)) \qquad (2.3)$$

分类的损失函数一般选择指数函数或对数函数，这里假设损失函数为对数函数，学习器的损失函数是

$$L(\theta) = \log(1 + \exp(-y_i \cdot \hat{y}_i^{(t)})) \qquad (2.4)$$

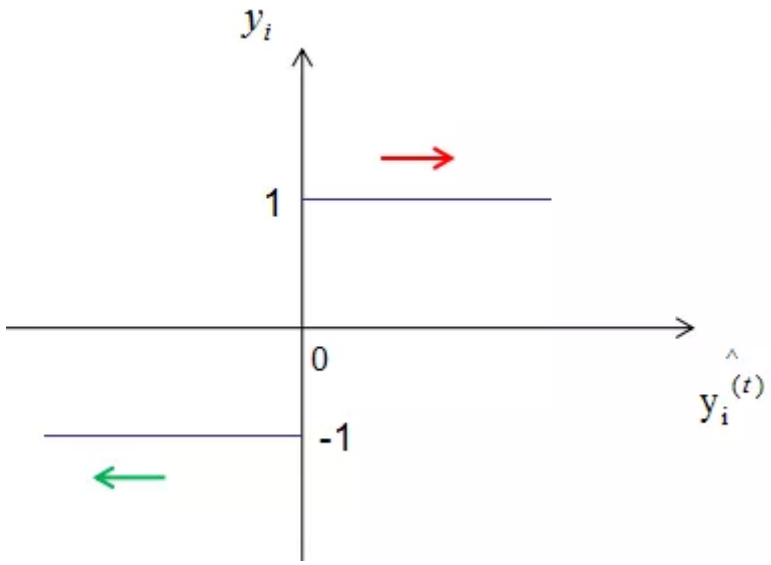
若实际输出结果yi=1，则：

$$L(\theta) = \log(1 + \exp(-\hat{y}_i^{(t)})) \qquad (2.5)$$

求（2.5）式对 $\hat{y}_i^{(t)}$ 的梯度，得：

$$-\frac{e^{-\hat{y}_i^{(t)}}}{1 + e^{-\hat{y}_i^{(t)}}} \qquad (2.6)$$

负梯度方向是损失函数下降最快的方向，（2.6）式取反的值大于0，因此弱学习器是往增大 $\hat{y}_i^{(t)}$ 的方向迭代的，图形表示为：



如上图，当样本的实际标记 yi 是 1 时，模型输出结果 $\hat{y}_i^{(t)}$ 随着迭代次数的增加而增加（红线箭头），模型的损失函数相应的减小；当样本的实际标记 yi 是 -1时，模型输出结果 $\hat{y}_i^{(t)}$ 随着迭代次数的增加而减小（红线箭头），模型的损失函数相应的减小 。这就是加法模型的原理所在，通过多次的迭代达到减小损失函数的目的。

小结：Boosting方法把每个弱学习器的输出看成是连续值，使得损失函数是个连续值，因此可以通过弱学习器的迭代达到优化模型的目的，这也是集成学习法加法模型的原理所在 。

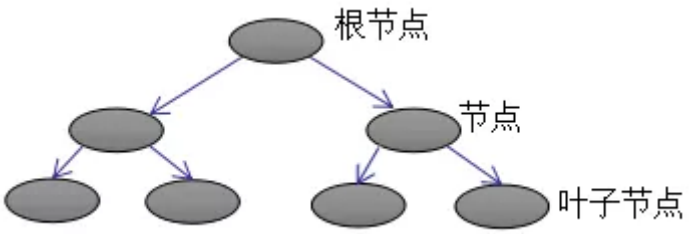
XGBoost算法的目标函数推导

目标函数，即损失函数，通过最小化损失函数来构建最优模型，由第一节可知， 损失函数应加上表示模型复杂度的正则项，且XGBoost对应的模型包含了多个CART树，因此，模型的目标函数为：

$$obj(\theta) = \sum_{i}^{n} L(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \qquad (3.1)$$

（3.1）式是正则化的损失函数，等式右边第一部分是模型的训练误差，第二部分是正则化项，这里的正则化项是K棵树的正则化项相加而来的。

CART树的介绍：



上图为第K棵CART树，确定一棵CART树需要确定两部分，第一部分就是树的结构，这个结构将输入样本映射到一个确定的叶子节点上，记为 $f_k(x)$ 。第二部分就是各个叶子节点的值，q(x)表示输出的叶子节点序号，$w_{q(x)}$ 表示对应叶子节点序号的值。由定义得：

$$f_k(x) = w_{q(x)} \qquad (3.2)$$

树的复杂度定义

XGBoost法对应的模型包含了多棵cart树，定义每棵树的复杂度：

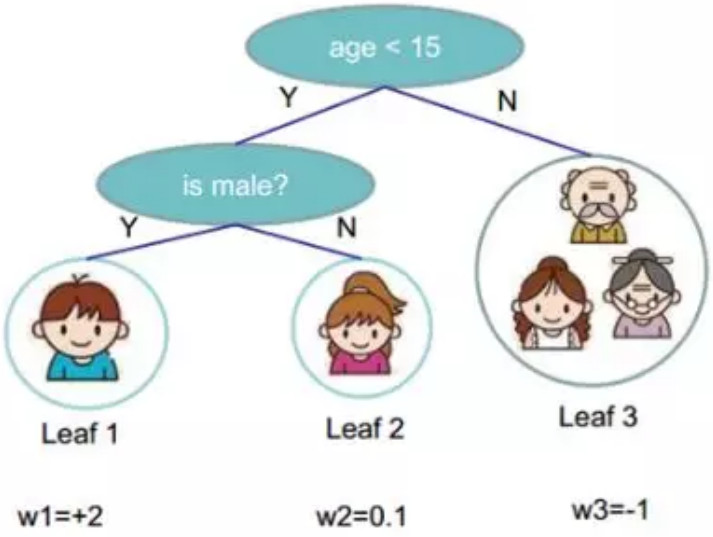
$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \| w \|^2 \qquad (3.4)$$

其中T为叶子节点的个数，||w||为叶子节点向量的模 。γ表示节点切分的难度，λ表示L2正则化系数。

如下例树的复杂度表示：

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

Number of leaves　　L2 norm of leaf scores



$$\Omega = \gamma 3 + \frac{1}{2}\lambda(4 + 0.01 + 1)$$

目标函数推导

根据（3.1）式，共进行t次迭代的学习模型的目标函数为：

$$obj^{(t)} = \sum_{i=1}^{n} L(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{t} \Omega(f_k)$$

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

由前向分布算法可知，前t - 1棵树的结构为常数

$$obj^{(t)} == \sum_{i=1}^{n} L(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + Cons\tan t \qquad (3.5)$$

泰勒公式的二阶导近似表示：

$$f(x_0+\Delta x)\approx f(x_0)+f'(x_0)\cdot(\Delta x)+\frac{f''(x_0)}{2}\cdot(\Delta x)^2 \qquad (3.6)$$

令$f_t(x_i)$为Δx，则（3.5）式的二阶近似展开：

$$obj^{(t)}\approx\sum_{i=1}^{n}[L(y_i,\hat{y}_i^{(t-1)})+g_i\cdot f_t(x_i)+\frac{1}{2}h_i\cdot f_t(x_i)^2]+\Omega(f_t)+Cons\tan t \qquad (3.7)$$

$\because \quad L(y_i,\hat{y}_i^{(t-1)})$为常数

$$\therefore \quad obj^{(t)}=\sum_{i=1}^{n}[g_i\cdot f_t(x_i)+\frac{1}{2}h_i\cdot f_t(x_i)^2]+\Omega(f_t)+Cons\tan t \qquad (3.8)$$

其中：

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

$$h_i=\partial^2_{\hat{y}^{(t-1)}}l(y_i,\hat{y}^{(t-1)})$$

$l(y_i,\hat{y}^{(t-1)})$表示前t-1棵树组成的学习模型的预测误差，gi和hi分别表示预测误差对当前模型的一阶导和二阶导，当前模型往预测误差减小的方向进行迭代。

忽略（3.8）式常数项，并结合（3.4）式，得：

$$obj^{(t)}=\sum_{i=1}^{n}[g_i\cdot f_t(x_i)+\frac{1}{2}h_i\cdot f_t(x_i)^2]+\gamma T+\frac{1}{2}\lambda\sum_{j=1}^{T}w^2{}_j \qquad (3.9)$$

通过（3.2）式简化（3.9）式：

$$obj^{(t)}=\sum_{i=1}^{n}[g_i\cdot w_{q(x_i)}+\frac{1}{2}h_i\cdot w^2{}_{q(x_i)}]+\gamma T+\frac{1}{2}\lambda\sum_{j=1}^{T}w^2{}_j \qquad (3.10)$$

（3.10）式第一部分是对所有训练样本集进行累加，因为所有样本都是映射为树的叶子节点，我们换种思维，从叶子节点出发，对所有的叶子节点进行累加，得：

$$obj^{(t)}=\sum_{j=1}^{T}[(\sum_{i\in I_j}g_i)w_j+\frac{1}{2}(\sum_{i\in I_j}h_i+\lambda)w_j^2]+\gamma T \qquad (3.11)$$

令

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

Gj 表示映射为叶子节点 j 的所有输入样本的一阶导之和，同理，Hj表示二阶导之和。

得：

$$obj^{(t)}=\sum_{j=1}^{T}[G_jw_j+\frac{1}{2}(H_j+\lambda)w_j^2]+\gamma T \qquad (3.12)$$

对于第 t 棵CART树的某一个确定结构（可用q(x)表示），其叶子节点是相互独立的，Gj和Hj是确定量，因此，（3.12）可以看成是关于叶子节点的一元二次函数 。最小化（3.12）式，得：

$$w_j^* = -\frac{G_j}{H_j + \lambda} \qquad (3.13)$$

得到最终的目标函数：

$$obj^* = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T \qquad (3.14)$$

（3.14）也称为打分函数(scoring function)，它是衡量树结构好坏的标准，值越小，代表这样的结构越好 。我们用打分函数选择最佳切分点，从而构建CART树。

CART回归树的构建方法

上节推导得到的打分函数是衡量树结构好坏的标准，因此，可用打分函数来选择最佳切分点。首先确定样本特征的所有切分点，对每一个确定的切分点进行切分，切分好坏的标准如下：

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

Gain表示单节点obj*与切分后的两个节点的树obj*之差，遍历所有特征的切分点，找到最大Gain的切分点即是最佳分裂点，根据这种方法继续切分节点，得到CART树。若 γ 值设置的过大，则Gain为负，表示不切分该节点，因为切分后的树结构变差了。γ值越大，表示对切分后obj下降幅度要求越严，这个值可以在XGBoost中设定。

XGBoost与GDBT的区别

1. XGBoost生成CART树考虑了树的复杂度，GDBT未考虑，GDBT在树的剪枝步骤中考虑了树的复杂度。
2. XGBoost是拟合上一轮损失函数的二阶导展开，GDBT是拟合上一轮损失函数的一阶导展开，因此，XGBoost的准确性更高，且满足相同的训练效果，需要的迭代次数更少。
3. XGBoost与GDBT都是逐次迭代来提高模型性能，但是XGBoost在选取最佳切分点时可以开启多线程进行，大大提高了运行速度。

PS:本节只选取了与本文内容相关的几个区别。

参考

陈天奇 《XGBoost:A Scalable Tree Boosting System》

李航 《统计学习方法》

推荐阅读

梯度提升树算法原理小结

比较全面的AdaBoost算法总结

比较全面的随机森林算法总结

集成学习原理总结

-END-

长按二维码关注

机器学习算法那些事
微信：beautifulife244

砥砺前行 不忘初心