GBDT梯度提升迭代决策树

1. 前言

在机器学习的有监督学习算法中，我们的目标是学习出一个稳定的且在各个方面表现都较好的模型，但实际情况往往不这么理想，有时我们只能得到多个有偏好的模型（弱监督模型，在某些方面表现的比较好）。集成学习就是组合这里的多个弱监督模型以期得到一个更好更全面的强监督模型，集成学习潜在的思想是即便某一个弱分类器得到了错误的预测，其他的弱分类器也可以将错误纠正回来。

集成方法是将几种机器学习技术组合成一个预测模型的元算法，以达到减小方差（bagging）、偏差（boosting）或改进预测（stacking）的效果。

Boosting其主要思想是将弱分类器组装成一个强分类器。在PAC（probably approximately correct，概率近似正确）学习框架下，则一定可以将弱分类器组装成一个强分类器。这族算法的工作机制类似：先从初始训练集训练出一个基学习器，再根据基学习器的表现对训练样本分布进行调整，使得先前基学习器做错的训练样本在后续受到更多关注, 然后基于调整后的样本分布来训练下一个基学习器；如此重复进行，直至基学习器数目达到事先指定的值T,最终将这T个基学习器进行加权结合.其中最著名的代表是AdaBoost算法。

GBDT (Gradient Boosting Decision Tree) 梯度提升迭代决策树。GBDT 也是 Boosting 算法的一种，但是和 AdaBoost 算法不同。区别如下：AdaBoost 算法是利用前一轮的弱学习器的误差来更新样本权重值，然后一轮一轮的迭代；GBDT 也是迭代，但是 GBDT 要求弱学习器必须是 CART 模型，而且 GBDT 在模型训练的时候，是要求模型预测的样本损失尽可能的小。

二、GBDT的负梯度拟合

GBDT的决策树有两种回归树和分类树，具体的决策树可以是CART或者C4.5生成。GBDT可以是学习得到一个强学习器f(x)，这样是通过梯度下降学习的，预测一个数值，这样的数值是连续的；也可以预测一个实数值，例如：10岁，20岁，5岁等，这样的数值是离散的；以上两种可以称为是回归时的功能。分类树是用于分类标签值，如晴天/阴天/雾/雨、用户性别、网页是否是垃圾页面。这里要强调的是，前者的结果加减是有意义的，如10岁+5岁-3岁=12岁，后者则无意义，如男+男+女=到底是男是女？要强调的是：不管是预测数值还是分类，在GBDT中都是学习残差的过程。

在GBDT的迭代中，假设我们前一轮迭代得到的强学习器是https://private.codecogs.com/gif.latex?f_%7Bt-1%7D%28x%29, 损失函数是L(y,{f_{t-1}(x)}), 本轮迭代的目标是找到一个CART回归树模型的弱学习器h_{t}(x)，让本轮的损失函数L(y, f_{t}(x)) = L(y,{f_{t-1}(x)+h_{t}(x)})最小。也就是说，本轮迭代找到决策树，要让样本的损失尽量变得更小。从上面的例子看这个思想还是蛮简单的，但是有个问题是这个损失的拟合不好度量，损失函数各种各样，怎么找到一种通用的拟合方法呢？Freidman提出了用损失函数的负梯度来拟合本轮损失的近似值，进而拟合一个CART回归树。第t轮的第i个样本的损失函数的负梯度表示为：

{r_{ti}}=-[\frac{\partial L(y_{i},f(x_{i}))}{\partial f(x_{i})}]_{f(x)=f_{t-1}(x)}

利用(x_{i},r_{ti}) (i=1,2,...,n),可以拟合一颗CART回归树，得到的第t棵回归树，其对应的叶节点区域{R_{tj}}, j=1,2,...,J。其中J为叶子节点的个数。针对每一个叶子节点里的样本，求出使损失函数最小，也就是拟合叶子节点最好的的输出值c_{tj}如下（注：这里的c代表的是未知变量，其求得的使损失函数L最小的值是c_{tj}）：

c_{tj} = argmin\sum_{x_{i}\epsilon {R_{tj}}}{L(y_{i},f_{t-1}(x_{i})}+c)

这样就得到了本轮的决策树拟合函数如下：

h_{t}(x)= \sum_{j=1}^{J}c_{tj}I(x\epsilon {R_{tj}})

经过本轮(第 t 轮)学习后得到的强学习器的表达式如下：

f_{t}(x)= f_{t-1}(x) + \sum_{j=1}^{J}c_{tj}I(x\epsilon {R_{tj}})

通过损失函数的负梯度来拟合，找到了一种通用的拟合损失误差的办法，这样无轮是分类问题还是回归问题，通过其损失函数的负梯度的拟合，就可以用GBDT来解决分类和回归问题。区别仅仅在于损失函数不同导致的负梯度不同而已。

三、GBDT回归算法

基于上面的思路，下面总结下GBDT的回归算法。此处不讨论分类算法，因为分类算法的输出是不连续的类别值，需要一些处理才能使用负梯度(定义损失函数时处理，我认为没啥区别，其本质是损失函数的设计，因为分类问题不能一步步拟合“残差”，以LR的对数似然函数作为损失函数来拟合“残差”，其本质是用类别的预测概率值来拟合正确的概率值)。

输入是训练集样本D={(x_{1},y_{1}),(x_{2},y_{2}),...,(x_{n},y_{n})}， 最大迭代次数T，损失函数L，输出是强学习器f(x)。

　1) 初始化弱学习器（这里初始化一个c的值，通常为所有样本点的平均值）

f_{0}(x)=argmin\sum_{i=1}^{n}L(y_{i},c)

　2) 对迭代轮数 t=1,2,...,T 有：

　a)对样本 i=1,2,...,n，计算负梯度

{r_{ti}}=-[\frac{\partial L(y_{i},f(x_{i}))}{\partial f(x_{i})}]_{f(x)=f_{t-1}(x)}

　b)利用(x_{i},r_{ti}) (i=1,2,...,n)，拟合一颗CART回归树，得到第 t 颗回归树，其对应的叶子节点区域为R_{tj}, j=1,2,...,J。其中J为回归树 t 的叶子节点的个数。

　　c) 对叶子区域j =1,2,3,...,J，计算最佳拟合值

c_{tj} = argmin\sum_{x_{i}\epsilon {R_{tj}}}{L(y_{i},f_{t-1}(x_{i})}+c)

d) 更新强学习器

f_{t}(x)= f_{t-1}(x) + \sum_{j=1}^{J}c_{tj}I(x\epsilon {R_{tj}})

3) 得到强学习器f(x)的表达式

f(x)=f_{T}(x)=f_{0}(x)+\sum_{t=1}^{T}\sum_{j=1}^{J}{c_{tj}}I(x\epsilon {R_{tj}})

四、GBDT常用损失函数

对于回归算法，常用损失函数有如下4种：

　　a)均方差，这个是最常见的回归损失函数

L(y,f(x))=(y-f(x))^2

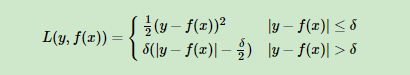
　　b)绝对损失，这个损失函数也很常见

L(y,f(x))=|y-f(x)|

对应负梯度误差为：

sign(y_{i}-f(x_{i}))

　　c)Huber损失，它是均方差和绝对损失的折衷产物，对于远离中心的异常点，采用绝对损失，而中心附近的点采用均方差。这个界限一般用分位数点度量。损失函数如下：



　　　　对应的负梯度误差为：

https://img-blog.csdnimg.cn/20181105202005788.png

d) 分位数损失。它对应的是分位数回归的损失函数，表达式为 https://img-blog.csdnimg.cn/2018110520202532.png

　其中θ为分位数，需要我们在回归前指定。对应的负梯度误差为：

https://img-blog.csdnimg.cn/2018110520204515.png

　　　　对于Huber损失和分位数损失，主要用于健壮回归，也就是减少异常点对损失函数的影响。

五、GBDT算法优缺点

优点：

预测精度高。

适合低维数据。

能处理非线性数据。

可以灵活处理各种类型的数据，包括连续值和离散值。

在相对少的调参时间情况下，预测的准备率也可以比较高。这个是相对SVM来说的。

使用一些健壮的损失函数，对异常值的鲁棒性非常强。比如 Huber损失函数和Quantile损失函数。

缺点：

由于弱学习器之间存在依赖关系，难以并行训练数据。

不过可以通过自采样的SGBT来达到部分并行。

如果数据维度较高时会加大算法的计算复杂度。