刘建平Pinard

十年码农，对数学统计学，数据挖掘，机器学习，大数据平台，大数据平台应用开发，大数据可视化感兴趣。

梯度提升树(GBDT)原理小结

在集成学习之Adaboost算法原理小结中，我们对Boosting家族的Adaboost算法做了总结，本文就对Boosting家族中另一个重要的算法梯度提升树(Gradient Boosting Decison Tree, 以下简称GBDT)做一个总结。GBDT有很多简称，有GBT（Gradient Boosting Tree），GTB（Gradient Tree Boosting ）， GBRT（Gradient Boosting Regression Tree），MART(Multiple Additive Regression Tree)，其实都是指的同一种算法，本文统一简称GBDT。GBDT在BAT大厂中也有广泛的应用，假如要选择3个最重要的机器学习算法的话，个人认为GBDT应该占一席之地。

1. GBDT概述

GBDT也是集成学习Boosting家族的成员，但是却和传统的Adaboost有很大的不同。回顾下Adaboost，我们是利用前一轮迭代弱学习器的误差率来更新训练集的权重，这样一轮轮的迭代下去。GBDT也是迭代，使用了前向分布算法，但是弱学习器限定了只能使用CART回归树模型，同时迭代思路和Adaboost也有所不同。

在GBDT的迭代中，假设我们前一轮迭代得到的强学习器是$f_{t-1}(x)$, 损失函数是$L(y, f_{t-1}(x))$, 我们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。

GBDT的思想可以用一个通俗的例子解释，假如有个人30岁，我们首先用20岁去拟合，发现损失有10岁，这时我们用6岁去拟合剩下的损失，发现差距还有4岁，第三轮我们用3岁拟合剩下的差距，差距就只有一岁了。如果我们的迭代轮数还没有完，可以继续迭代下面，每一轮迭代，拟合的岁数误差都会减小。

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?

2. GBDT的负梯度拟合

在上一节中，我们介绍了GBDT的基本思路，但是没有解决损失函数拟合方法的问题。针对这个问题，大牛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

$$r_{ti} = -\left[\frac{\partial L(y_i, f(x_i)))}{\partial f(x_i)}\right]_{f(x)=f_{t-1}(x)}$$

利用$(x_i, r_{ti})\ \ (i = 1, 2, .. m)$,我们可以拟合一颗CART回归树，得到了第t颗回归树，其对应的叶节点区域$R_{tj}, j = 1, 2, \ldots, J$。其中J为叶子节点的个数。

针对每一个叶子节点里的样本，我们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:

$$c_{tj} = \underbrace{arg\ min}_{c} \sum_{x_i \in R_{tj}} L(y_i, f_{t-1}(x_i) + c)$$

这样我们就得到了本轮的决策树拟合函数如下:

$$h_t(x) = \sum_{j=1}^{J} c_{tj} I(x \in R_{tj})$$

从而本轮最终得到的强学习器的表达式如下:

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

通过损失函数的负梯度来拟合，我们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，我们通过其损失函数的负梯度的拟合，就可以用GBDT来解决我们的分类回归问题。区别仅仅在于损失函数不同导致的负梯度不同而已。

3. GBDT回归算法

好了，有了上面的思路，下面我们总结下GBDT的回归算法。为什么没有加上分类算法一起？那是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，我们在下一节讲。

输入是训练集样本$T = \{(x, y_1), (x_2, y_2), \ldots (x_m, y_m)\}$， 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

$$f_0(x) = \underbrace{arg\ min}_{c} \sum_{i=1}^{m} L(y_i, c)$$

2) 对迭代轮数t=1,2,...T有:

a)对样本i=1,2，...m，计算负梯度

$$r_{ti} = -\left[\frac{\partial L(y_i, f(x_i)))}{\partial f(x_i)}\right]_{f(x)=f_{t-1}(x)}$$

b)利用$(x_i, r_{ti})\ \ (i = 1, 2, ..m)$, 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$R_{tj}, j = 1, 2, \ldots, J$。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,..J,计算最佳拟合值

$$c_{tj} = \underbrace{arg\ min}_{c} \sum_{x_i \in R_{tj}} L(y_i, f_{t-1}(x_i) + c)$$

d) 更新强学习器

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

3) 得到强学习器f(x)的表达式

$$f(x) = f_T(x) = f_0(x) + \sum_{t=1}^{T} \sum_{j=1}^{J} c_{tj} I(x \in R_{tj})$$

4. GBDT分类算法

这里我们再看看GBDT分类算法，GBDT的分类算法从思想上和GBDT的回归算法没有区别，但是由于样本输出不是连续的值，而是离散的类别，导致我们无法直接从输出类别去拟合类别输出的误差。

为了解决这个问题，主要有两个方法，一个是用指数损失函数，此时GBDT退化为Adaboost算法。另一种方法是用类似于逻辑回归的对数似然损失函数的方法。也就是说，我们用的是类别的预测概率值和真实概率值的差来拟合损失。本文仅讨论用对数似然损失函数的GBDT分类。而对于对数似然损失函数，我们又有二元分类和多元分类的区别。

4.1 二元GBDT分类算法

对于二元GBDT，如果用类似于逻辑回归的对数似然损失函数，则损失函数为：

$$L(y, f(x)) = log(1 + exp(-yf(x)))$$

其中$y \in \{-1, +1\}$。则此时的负梯度误差为

$$r_{ti} = -\left[\frac{\partial L(y, f(x_i)))}{\partial f(x_i)}\right]_{f(x)=f_{t-1}(x)} = y_i / (1 + exp(y_i f(x_i)))$$

对于生成的决策树，我们各个叶子节点的最佳负梯度拟合值为

$$c_{tj} = \underbrace{arg\ min}_{c} \sum_{x_i \in R_{tj}} log(1 + exp(-y_i(f_{t-1}(x_i) + c)))$$

由于上式比较难优化，我们一般使用近似值代替

$$c_{tj} = \sum_{x_i \in R_{tj}} r_{ti} \Big/ \sum_{x_i \in R_{tj}} |r_{ti}|(1 - |r_{ti}|)$$

除了负梯度计算和叶子节点的最佳负梯度拟合的线性搜索，二元GBDT分类和GBDT回归算法过程相同。

4.2 多元GBDT分类算法

多元GBDT要比二元GBDT复杂一些，对应的是多元逻辑回归和二元逻辑回归的复杂度差别。假设类别数为K，则此时我们的对数似然损失函数为：

$$L(y, f(x)) = -\sum_{k=1}^{K} y_k log\ p_k(x)$$

其中如果样本输出类别为k，则$y_k = 1$。第k类的概率$p_k(x)$的表达式为：

$$p_k(x) = exp(f_k(x)) \Big/ \sum_{l=1}^{K} exp(f_l(x))$$

集合上两式，我们可以计算出第t轮的第i个样本对应类别l的负梯度误差为

$$r_{til} = -\left[\frac{\partial L(y_i, f(x_i)))}{\partial f(x_i)}\right]_{f_k(x)=f_{l,t-1}(x)} = y_{il} - p_{l,t-1}(x_i)$$

观察上式可以看出，其实这里的误差就是样本i对应类别l的真实概率和$t-1$轮预测概率的差值。

对于生成的决策树，我们各个叶子节点的最佳负梯度拟合值为

$$c_{tjl} = \underbrace{arg\ min}_{c_{jl}} \sum_{i=0}^{m} \sum_{k=1}^{K} L(y_k, f_{t-1,l}(x) + \sum_{j=0}^{J} c_{jl} I(x_i \in R_{tjl}))$$

由于上式比较难优化，我们一般使用近似值代替

$$c_{tjl} = \frac{K-1}{K} \frac{\sum\limits_{x_i \in R_{tjl}} r_{til}}{\sum\limits_{x_i \in R_{til}} |r_{til}|(1 - |r_{til}|)}$$

除了负梯度计算和叶子节点的最佳负梯度拟合的线性搜索，多元GBDT分类和二元GBDT分类以及GBDT回归算法过程相同。

5. GBDT常用损失函数

这里我们再对常用的GBDT损失函数做一个总结。

对于分类算法，其损失函数一般有对数损失函数和指数损失函数两种：

a) 如果是指数损失函数，则损失函数表达式为

$$L(y, f(x)) = exp(-yf(x))$$

其负梯度计算和叶子节点的最佳负梯度拟合参见Adaboost原理篇。

b) 如果是对数损失函数，分为二元分类和多元分类两种，参见4.1节和4.2节。

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了

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

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：

$$L(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & |y - f(x)| \le \delta \\ \delta(|y - f(x)| - \frac{\delta}{2}) & |y - f(x)| > \delta \end{cases}$$

对应的负梯度误差为：

$$r(y_i, f(x_i)) = \begin{cases} y_i - f(x_i) & |y_i - f(x_i)| \le \delta \\ \delta sign(y_i - f(x_i)) & |y_i - f(x_i)| > \delta \end{cases}$$

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

$$L(y, f(x)) = \sum_{y \ge f(x)} \theta |y - f(x)| + \sum_{y < f(x)} (1 - \theta)|y - f(x)|$$

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：

$$r(y_i, f(x_i)) = \begin{cases} \theta & y_i \ge f(x_i) \\ \theta - 1 & y_i < f(x_i) \end{cases}$$

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。

6. GBDT的正则化

和Adaboost一样，我们也需要对GBDT进行正则化，防止过拟合。GBDT的正则化主要有三种方式。

第一种是和Adaboost类似的正则化项，即步长(learning rate)。定义为ν,对于前面的弱学习器的迭代

$$f_k(x) = f_{k-1}(x) + h_k(x)$$

如果我们加上了正则化项，则有

$$f_k(x) = f_{k-1}(x) + \nu h_k(x)$$

ν的取值范围为$0 < \nu \le 1$。对于同样的训练集学习效果，较小的ν意味着我们需要更多的弱学习器的迭代次数。通常我们用步长和迭代最大次数一起来决定算法的拟合效果。

第二种正则化的方式是通过子采样比例（subsample）。取值为(0,1]。注意这里的子采样和随机森林不一样，随机森林使用的是放回抽样，而这里是不放回抽样。如果取值为1，则全部样本都使用，等于没有使用子采样。如果取值小于1，则只有一部分样本会去做GBDT的决策树拟合。选择小于1的比例可以减少方差，即防止过拟合，但是会增加样本拟合的偏差，因此取值不能太低。推荐在[0.5, 0.8]之间。

使用了子采样的GBDT有时也称作随机梯度提升树(Stochastic Gradient Boosting Tree, SGBT)。由于使用了子采样，程序可以通过采样分发到不同的任务去做boosting的迭代过程，最后形成新树，从而减少弱学习器难以并行学习的弱点。

第三种是对于弱学习器即CART回归树进行正则化剪枝。在决策树原理篇里我们已经讲过，这里就不重复了。

7. GBDT小结

GBDT终于讲完了，GDBT本身并不复杂，不过要吃透的话需要对集成学习的原理，决策树原理和各种损失函树有一定的了解。由于GBDT的卓越性能，只要是研究机器学习都应该掌握这个算法，包括背后的原理和应用调参方法。目前GBDT的算法比较好的库是xgboost。当然scikit-learn也可以。

最后总结下GBDT的优缺点。

GBDT主要的优点有：

1) 可以灵活处理各种类型的数据，包括连续值和离散值。

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

3）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。

GBDT的主要缺点有：

1)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。

以上就是GBDT的原理总结，后面会讲GBDT的scikit-learn调参，敬请期待。

（欢迎转载，转载请注明出处。欢迎沟通交流： liujianping-ok@163.com）

分类: 0081. 机器学习

标签: 集成学习

刘建平Pinard
关注 - 16
粉丝 - 6704
我在关注他 取消关注

48 0

« 上一篇： scikit-learn Adaboost类库使用小结
» 下一篇： scikit-learn 梯度提升树(GBDT)调参小结

posted @ 2016-12-07 19:59 刘建平Pinard 阅读(240083) 评论(538) 编辑 收藏

< Prev 1 … 5 6 7 8 9 10 11

评论列表

#501楼 [楼主] 2020-04-25 12:01 刘建平Pinard 回复 引用

@鱼水千行
你好，其实GBDT分类里面也使用损失函数计算一阶二阶导了，当然树的分裂依据没有直接使用。
用损失函数计算一阶二阶导参见<XGBoost算法原理小结>里的第三节：https://www.cnblogs.com/pinard/p/10979808.html

支持(1) 反对(0)

#502楼 2020-04-25 16:46 鱼水千行 回复 引用

非常感谢您的解答！
还想再问问您对xgboost之类的集成算法应用于面板数据是否有需要改进的地方，我的想法是：

1. 在时间维度，数据的顺序也是蕴含信息的，所以似乎不适合用随机抽取样本的方式生成多个树；
2. 一般时间序列的数据是会自相关的，所以可以根据情况将因变量y的滞后一阶或多阶作为特征；
3. xgboost是可以自定义损失函数的，而距今越近的数据价值越高，距今越远的数据价值越低，所以可以将损失函数指数平滑化来体现这一特点，但会不会和第2条的滞后项产生重复的效果？
4. 有没有必要做成动态的模型？比如以7年为窗口期，每预测一天的因变量就用前7年的数据拟合一个xgboost模型，这样每天的预测都对应着不同的模型。还是说用7年的数据拟合一个模型，之后的每一天都用同一个模型预测就好？

问题太多，不胜叨扰，但还是希望能听听您的意见

支持(0) 反对(0)

#503楼 [楼主] 2020-04-26 09:08 刘建平Pinard 回复 引用

@鱼水千行
你好！
1， XGBoost之类的算法假设训练样本之间是完全独立的，你加入了顺序，这块特征不太适合直接在XGBoost来建模，你可以考虑每个样本加了2个（组）特征，第一组是上个时间点的特征信息，第二组是下个时间点的特征信息。这样所有的样本就都还是独立的，同时你也有上下文时间信息了。
2，可以的，和我在第一天的思路相近，你可以考虑。
3，这个损失函数指数平滑化是可以的，不会产生重复效果，只能说加强了这块信息的价值。你也可以通过加入样本权重sample_weight来处理，每个样本的sample_weight和它的时间远近有关。
4，这个看你的业务特性和业务需要，如果你觉得这样是合理的，就可以尝试去做了，2个方法都有可取之处。我这边实际项目里，会考虑窗口期，即你的第一种方法，因为随时间推移，数据模型会发生变化的。

支持(0) 反对(0)

#504楼 2020-04-26 10:09 鱼水千行 回复 引用

@刘建平Pinard
明白了，非常感谢您的指导！！！

支持(0) 反对(0)

#505楼 2020-04-27 20:42 aspire_zgq 回复 引用

你好，请问一下GBDT在构建模型的时候因变量y可以是多维的吗？

支持(0) 反对(0)

#506楼 [楼主] 2020-04-28 08:55 刘建平Pinard 回复 引用

@aspire_zgq
你好，sklearn的GBDT是不支持多因变量的模型的。你可以对每个因变量单独跑GBDT，建立多个GBDT模型。

支持(0) 反对(0)

#507楼 2020-04-28 09:03 aspire_zgq 回复 引用

@刘建平Pinard
好的，谢谢刘老师

支持(0) 反对(0)

#508楼 2020-05-04 23:01 呀哈nemo 回复 引用

楼主您好，有一个问题想请教一下一直没有看懂。就是在计算负梯度之后，基于样本和负梯度已经拟合了一个决策树模型了，相当于有一个确定的c了，后为什么会有很多个c进行最佳拟合值的计算呢?

支持(0) 反对(0)

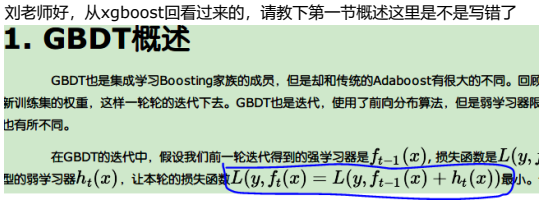
#509楼 [楼主] 2020-05-06 09:00 刘建平Pinard 回复 引用

@呀哈nemo
你好！ 关键点在于第t论迭代，得到了第t颗回归树后，我们的目的是期望全局的损失最小，而不是第t颗决策树拟合后第t轮负梯度的误差最小。注意到第t轮负梯度和全局的误差并不相同。
类似的问题在前面也讲过很多次，比如482/484楼，你可以翻一下。

支持(0) 反对(0)

#510楼 2020-06-07 13:43 空无名 回复 引用

刘老师好，从xgboost回看过来的，请教下第一节概述这里是不是写错了



等号左边少打了个半括号。

支持(0) 反对(0)

#511楼 [楼主] 2020-06-08 09:13 刘建平Pinard 回复 引用

@空无名
你好，感谢指正，原文已修改。

支持(0) 反对(0)

#512楼 2020-07-12 18:25 呀哈nemo 回复 引用

刘老师好，相关于采样的问题请教一下。gbdt的采样是不放回采样，是指第一次训练之前就做一次采样吗，之后的迭代实际上都是基于第一次训练的样本？还是说每一次迭代都是一次新的采样，每一次都会上次未选中20%样本的80%？如果每一次都是新的采样，那在上一轮训练的结果是用上一轮的模型进行预测再计算误差吗？同样的，对于rf模型，放回采样模型是指每一次弱学习器迭代都要重新抽样吗？那如果上一轮没有出现的样本这次出现了，权重还是1/m?谢谢~

支持(0) 反对(0)

#513楼 [楼主] 2020-07-13 09:58 刘建平Pinard 回复 引用

@呀哈nemo
你好!
对于GBDT，每一次迭代都是一次新的不放回采样，不过是重新从全部的训练集中开始采样，而不是说“，每一次都会上次未选中20%样本的80%”。此时所有样本的误差基于上一轮预测的结果来计算。
对于rf模型，放回采样也是指每一次弱学习器迭代都要重新放回采样，也是从全部的训练集中开始采样，所以如果上一轮没有出现的样本这次出现了，，每次被采集到的概率也还是1/m。

支持(0) 反对(0)

#514楼 2020-07-13 21:38 呀哈nemo 回复 引用

@刘建平Pinard
明白了 谢谢刘老师

支持(0) 反对(0)

#515楼 2020-07-20 21:29 呀哈nemo 回复 引用

刘老师您好，关于模型归一化标准化的问题想请教一下，gbdt包括xgb模型需要做归一化或者标准化处理吗？我看相关的介绍是归一化可以提高迭代的速度，会对准确性有一定的影响吗？因为做决策树的时候不需要归一化，gbdt的弱学习器也是决策树，所以不会产生影响，是这样吗？非常感谢您的解答

支持(0) 反对(0)

#516楼 [楼主] 2020-07-21 09:08 刘建平Pinard 回复 引用

@呀哈nemo
你好，gbdt包括xgb模型都需要做归一化的，归一化以后模型的准确率一般会有一定程度的提升。
决策树这样的的确可以不做归一化。但是GBDT一般还是要做的，毕竟它不是单纯的决策树模型，需要做梯度的迭代更新。

支持(0) 反对(0)

#517楼 2020-07-21 12:08 呀哈nemo 回复 引用

@刘建平Pinard
谢谢刘老师，还是想问下，基于二分类的问题，每一步回归拟合的残差也只是基于y值和若学习器拟合的结果，和模型的特征变量没有直接的关系？

支持(0) 反对(0)

#518楼 [楼主] 2020-07-22 08:48 刘建平Pinard 回复 引用

@呀哈nemo
你好，对于GBDT，虽然每一步回归拟合的残差也只是基于y值和若学习器拟合的结果，但是他的本质还是基于泰勒公式的展开，与导数（梯度）有关系，一般与导数（梯度）有关系的算法对应的数据特征都需要做归一化。

支持(0) 反对(0)

#519楼 2020-08-12 20:21 明天，加油 回复 引用

@刘建平Pinard
老师，您好，关于归一化这里我也想问下，一般来说需要计算梯度的模型都需要做归一化，但是在GBDT里面计算梯度并不是对特征求的梯度呀，是损失函数对当前模型求的负梯度，与特征没有关系，这样还需要做归一化吗?

支持(0) 反对(0)

#520楼 [楼主] 2020-08-13 09:28 刘建平Pinard 回复 引用

@明天，加油
你好，GBDT的归一化如你所说，从理论上不是必须的，但是归一化了效果很多时候算法效果还是有提升的，所以大部分做GBDT落地的还是会对特征做归一化的。也就是我518楼说的，基本至要涉及梯度优化的，我们做算法落地都会归一化。

支持(0) 反对(0)

#521楼 2020-08-14 15:40 埋头学习，不问世事。 回复 引用

你好，在网上找到的对数损失函数都是强调标签y∈{-1,1}，包括老师你的这篇文章也是。可是我们平时很多任务都是标签y∈{0,1}，这时的损失函数是怎样的？用sklearn的GBDT，它能自动根据负例是0或者-1来选用损失函数吗?

支持(0) 反对(0)

#522楼 [楼主] 2020-08-17 09:23 刘建平Pinard 回复 引用

@埋头学习，不问世事。
你好，选择将输出标签设置为（0,1）或者（-1,1）是根据你的损失函数决定的，一般指数损失函数都是用（-1,1），而对数损失函数用（0,1）的比较多。
而输出标签是0还是-1是你自己可以定的，比如把所有样本的输出从0替换成-1，也是可以的。
在sklearn的算法库里面，你只要输出是确定的几个类别标签就可以了，不需要自己去搞成-1,1或者0,1。算法库自己会自动搞定的。
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html#sklearn.ensemble.GradientBoostingClassifier.fit
y，array-like of shape (n_samples,)
Target values (strings or integers in classification, real numbers in regression) For classification, labels must correspond to classes.

支持(0) 反对(0)

#523楼 2020-08-25 23:11 hanyuyyy 回复 引用

刘老师，您好，我其实一直对GBDT中拟合负梯度心存疑惑，在树生长的过程中首先计算出每个样本损失函数的负梯度，是要进行一阶泰勒展开以构造出关于预测值的函数吗（像Xgboost那样）？我疑惑的点在于，在第C步，计算最佳拟合值的这一步目标函数也是关于预测值的单变量函数啊，那负梯度发挥了什么作用，为什么是拟合负梯度，谢谢老师?

支持(0) 反对(0)

#524楼 [楼主] 2020-08-26 09:25 刘建平Pinard 回复 引用

@hanyuyyy
你好，对的，就是一阶泰勒展开进行拟合，和XGBoost类似，只不过是二阶泰勒展开拟合。
第C步只是为了修正，因为我们的问题并不是为了当前迭代的那颗树损失最小，而是整体的损失最小，所以需要再做一次线性搜索优化一下。
也就是负梯度的拟合是基于泰勒公式展开希望拟合当前的决策树。而线性搜索时是进行修正，使拟合的结果对全局损失更小。

支持(0) 反对(0)

#525楼 2020-08-26 10:04 hanyuyyy 回复 引用

@刘建平Pinard
老师，那一阶泰勒展开后优化的目标不是整体损失最小吗？像Xgboost那样，Xgboost中好像没有修正这一步，是因为用泰勒展开来模拟损失目标函数有偏差所以修正吗?

支持(0) 反对(0)

#526楼 2020-08-26 10:13 hanyuyyy 回复 引用

@刘建平Pinard
老师，我的另一个理解是，首先计算损失函数对每个样本当前预测值的负梯度，本轮决策树拟合值逼近的目标就是这个负梯度，构建出决策树，由于我们的目标是全局损失最小，所以需要加第C步进行修正。不知道哪种理解是正确的。希望老师指导。

支持(0) 反对(0)

#527楼 [楼主] 2020-08-27 09:14 刘建平Pinard 回复 引用

@hanyuyyy
你好!
525楼：Xgboost中没有修正这一步，是因为它在决策树分裂的时候每一次分裂都在考虑要全局的损失函数最小，而GBDT还是按普通的CART树拟合当前负梯度，所以GBDT需要修正，而XGBoost不需要。
526楼：你这里的理解没问题。

支持(0) 反对(0)

#528楼 2020-09-23 11:19 niunai96 回复 引用

老师您好:
我想问一下，如果数据标签值都在0-1000范围之内，gbdt做回归的值能否超过1000吗？因为gbdt做回归一般都是用均方差损失函数，是不断的对残差求均值的过程，我个人感觉是不会超过的，但是不太相信自己，还想和您确认一下，谢谢您!

支持(0) 反对(0)

#529楼 [楼主] 2020-09-24 09:37 刘建平Pinard 回复 引用

@niunai96
你好，你说的是回归问题的GBDT模型是吧，如果你的训练数据都是0-1000的输出，做预测的时候的输出是可以超过1000的，看你的测试集的数据分布情况了。
虽然gbdt做回归是均方误差残差拟合，但是你叶子节点的各个区域取值有大有小。如果新的测试样本刚好在各个树的叶子区域都是取大值，那么是可以超过训练集里最大的输出值的。

支持(0) 反对(0)

#530楼 2020-09-24 12:44 niunai96 回复 引用

@刘建平Pinard
谢谢您，那如果是随机森林也可以超过吗，随机森林是基于bagging的思想，回归问题最后取平均，那么测试数据是不是不会超过1000

支持(0) 反对(0)

#531楼 [楼主] 2020-09-25 09:19 刘建平Pinard 回复 引用

@niunai96
你好，一样可以超过1000的，看你每颗决策树里叶子节点的取值情况，，如果有某个测试样本，在各个决策树里的取值都是最大的叶子节点，那么就算是平均后也可能超出训练集的输出最大值。

支持(0) 反对(0)

#532楼 2020-09-27 09:32 niunai96 回复 引用

老师您好:
我想问下梯度提升树里的这个梯度是什么，是不是就是损失函数对前一轮强学习器的负梯度？谢谢老师

支持(0) 反对(0)

#533楼 [楼主] 2020-09-27 09:38 刘建平Pinard 回复 引用

@niunai96
你好，对的，就是损失函数对前一轮强学习器的负梯度。

支持(0) 反对(0)

#534楼 2020-10-17 22:24 子晔 回复 引用

刘老师您好，我想请问一下，gbdt的损失函数在实际应用中要如何选择呢？会根据什么条件，比如数据量之类的吗，谢谢您

支持(0) 反对(0)

#535楼 [楼主] 2020-10-19 09:35 刘建平Pinard 回复 引用

@子晔
你好，一般损失函数的选择不会太复杂，也不用考虑那么多，在没特殊要求的情况下，回归你选择MSE，分类你选择softmax（sigmoid）就可以了。

支持(0) 反对(0)

#536楼 2020-10-20 09:58 王大大·· 回复 引用

刘老师您好，我对您513楼关于采样的描述有疑问，随机森林和GBDT都是"重新从全部的训练集中开始采样"，那有什么区别呢，这样的话不都是有放回的采样吗

支持(0) 反对(0)

#537楼 [楼主] 2020-10-20 10:41 刘建平Pinard 回复 引用

@王大大··
你好!
随机森林和GBDT都是"重新从全部的训练集中开始采样"===》 这里说的是对一个新的弱学习器决策树来说的，所谓"重新"是针对上一个弱学习器来说的，上一个弱学习器采样过哪些样本我们不管，在当前弱学习器，我们从全部的训练集中开始采样。
"放回"和"不放回"是针对当前的弱学习器采样来说的，当前弱学习器最开始我们从全部的训练集中开始采样，此时我们会先采样到第一个样本，如果是"放回"采样，那么当前这个弱学习器的采样还可以在后面采样到这第一个样本，如果是"不放回"，那么当前这个弱学习器的采样后面就不会再采样到这个样本了。

支持(0) 反对(0)

#538楼 2020-10-20 10:49 王大大·· 回复 引用

@刘建平Pinard
谢谢您，我转过弯来了

支持(0) 反对(0)

< Prev 1 … 5 6 7 8 9 10 11

刷新评论 刷新页面 返回顶部

发表评论

编辑 预览

支持 Markdown

提交评论 退出 订阅评论

[Ctrl+Enter快捷键提交]

相关博文：
· CTR预估-GBDT与LR实现
· 平衡树B树B+树红黑树
· 树--二叉树
· B树和B+树
· 树
» 更多推荐...