# 机器学习算法总结(四)——GBDT与XGBOOST

Boosting方法实际上是采用加法模型与前向分布算法。在上一篇提到的Adaboost算法也可以用加法模型和前向分布算法来表示。以决策树为基学习器的提升方法称为提升树（Boosting Tree）。对分类问题决策树是CART分类树，对回归问题决策树是CART回归树。

## 1、前向分布算法

引入加法模型

$$f(x)=\sum_{m=1}^{M}\beta_m b(x;\gamma_m)$$

在给定了训练数据和损失函数$L(y,f(x))$的条件下，可以通过损失函数最小化来学习加法模型

$$\min_{\beta_m,\gamma_m}\sum_{i=1}^{N}L\left(y_i,\sum_{m=1}^{M}\beta_m b(x_i;\gamma_m)\right)$$

然而对于这个问题是个很复杂的优化问题，而且要训练的参数非常的多，前向分布算法的提出就是为了解决模型的优化问题，其核心思想是因为加法模型是由多各模型相加在一起的，而且在Boosting中模型之间又是有先后顺序的，因此可以在执行每一步加法的时候对模型进行优化，那么每一步只需要学习一个模型和一个参数，通过这种方式来逐步逼近全局最优，每一步优化的损失函数：

$$\min_{\beta,\gamma}\sum_{i=1}^{N}L\left(y_i,\beta b(x_i;\gamma)\right)$$

具体算法流程如下：

1）初始化$f_0(x)=0$;

2）第m次迭代时，极小化损失函数

$$(\beta_m,\gamma_m)=\arg\min_{\beta,\gamma}\sum_{i=1}^{N}L\left(y_i,f_{m-1}(x_i)+\beta b(x_i;\gamma)\right)$$

3）更新模型，则$f_m (x)$：

$$f_m(x)=f_{m-1}(x)+\beta_m b(x;\gamma_m)$$

4）得到最终的加法模型

$$f(x)=f_M(x)=\sum_{m=1}^{M}\beta_m b(x;\gamma_m)$$

**Adaboost算法也可以用前向分布算法来描述，在这里输入的数据集是带有权重分布的数据集，损失函数是指数损失函数。**

## 2、GBDT算法

GBDT是梯度提升决策树（Gradient Boosting Decision Tree）的简称，GBDT可以说是最好的机器学习算法之一。**GBDT分类和回归时的基学习器都是CART回归树，因为是拟合残差的。**GBDT和Adaboost一样可以用前向分布算法来描述，不同之处在于Adaboost算法每次拟合基学习器时，输入的样本数据是不一样的（每一轮迭代时的样本权重不一致），因为Adaboost旨在重点关注上一轮分类错误的样本，GBDT算法在每一步迭代时是输出的值不一样，本轮要拟合的输出值是之前的加法模型的预测值和真实值的差值（模型的残差，也称为损失）。用于一个简单的例子来说明GBDT，假如某人的年龄为30岁，第一次用20岁去拟合，发现损失还有10岁，第二次用6岁去拟合10岁，发现损失还有4岁，第三次用3岁去拟合4岁，依次下去直到损失在我们可接受范围内。

以平方误差损失函数的回归问题为例，来看看以损失来拟合是个什么样子，采用前向分布算法：

$$f_0(x)=0$$
$$f_m(x)=f_{m-1}(x)+T(x;\Theta_m),\quad m=1,2,\cdots,M$$
$$f_M(x)=\sum_{m=1}^{M}T(x;\Theta_m)$$

在第$m$次迭代时，我们要优化的损失函数：

$$\hat{\Theta}_m=\arg\min_{\Theta_m}\sum_{i=1}^{N}L(y_i,f_{m-1}(x_i)+T(x_i;\Theta_m))$$

此时我们采用平方误差损失函数为例：

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

则上面损失函数变为：

$$\begin{aligned}&L(y,f_{m-1}(x)+T(x;\Theta_m))\\&\quad=[y-f_{m-1}(x)-T(x;\Theta_m)]^2\\&\quad=[r-T(x;\Theta_m)]^2\end{aligned}$$

问题就成了对残差r的拟合了

然而对于大多数损失函数，却没那么容易直接获得模型的残差，针对该问题，大神Freidman提出了用损失函数的负梯度来拟合本轮损失的近似值，拟合一个回归树

$$-\left[\frac{\partial L(y,f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)}$$

关于GBDT一般损失函数的具体算法流程如下：

1）初始化$f_0(x)$：

$$f_0(x)=\arg\min_{c}\sum_{i=1}^{N}L(y_i,c)$$

2）第$m$次迭代时，计算当前要拟合的残差$r_{mi}$：

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

以$r_{mi}$为输出值，对$r_{mi}$拟合一个回归树（**此时只是确定了树的结构，但是还未确定叶子节点中的输出值**），然后通过最小化当前的损失函数，并求得每个叶子节点中的输出值$c_{mj}$，$j$表示第$j$个叶子节点

$$c_{mj}=\arg\min_{c}\sum_{x_i\in R_{mj}}L(y_i,f_{m-1}(x_i)+c)$$

更新当前的模型$f_m(x)$为：

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

3）依次迭代到我们设定的基学习器的个数$M$，得到最终的模型，其中$M$表示基学习器的个数，$J$表示叶子节点的个数

$$\hat{f}(x)=f_M(x)=\sum_{m=1}^{M}\sum_{j=1}^{J}c_{mj}I(x\in R_{mj})$$

GBDT算法提供了众多的可选择的损失函数，通过选择不同的损失函数可以用来处理分类、回归问题，比如用对数似然损失函数就可以处理分类问题。大概的总结下常用的损失函数：

1）对于分类问题可以选用指数损失函数、对数损失函数。

2）对于回归问题可以选用均方差损失函数、绝对损失函数。

3）另外还有huber损失函数和分位数损失函数，也是用于回归问题，可以增加回归问题的健壮性，可以减少异常点对损失函数的影响。

## 3、GBDT的正则化

在Adaboost中我们会对每个模型乘上一个弱化系数（正则化系数），减小每个模型对提升的贡献（**注意：这个系数和模型的权重不一样，是在权重上又乘以一个0,1之间的小数**），在GBDT中我们采用同样的策略，对于每个模型乘以一个系数λ (0 < λ ≤ 1)，降低每个

模型对拟合损失的贡献，这种方法也意味着我们需要更多的基学习器。

第二种是每次通过按比例（推荐[0.5, 0.8] 之间）随机抽取部分样本来训练模型，这种方法有点类似Bagging，可以减小方差，但同样会增加模型的偏差，可采用交叉验证选取，这种方式称为子采样。采用子采样的GBDT有时也称为随机梯度提升树（SGBT）。

第三种就是控制基学习器CART树的复杂度，可以采用剪枝正则化。

## 4、GBDT的优缺点

GBDT的主要优点：

1）可以灵活的处理各种类型的数据

2）预测的准确率高

3）使用了一些健壮的损失函数，如huber，可以很好的处理异常值

GBDT的缺点：

1）由于基学习器之间的依赖关系，难以并行化处理，不过可以通过子采样的SGBT来实现部分并行。

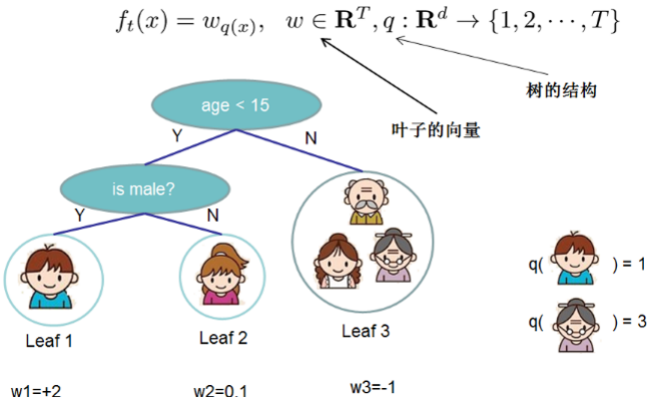
## 5、XGBoost算法

**事实上对于树模型为基学习器的集成方法在建模过程中可以分为两个步骤：一是确定树模型的结构，二是确定树模型的叶子节点中的输出值。**

### 5.1 定义树的复杂度

首先把树拆分成结构部分$q$和叶子节点输出值$w$，在这里$w$是一个向量，表示各叶子节点中的输出值。在这里就囊括了上面提到的两点，确定树结构$q$和叶子结点的输出值$w$。从下图中可以看出，$q(x)$实际上是确定输入值最终会落到哪个叶子节点上，而$w$将会给出相应的输出值。

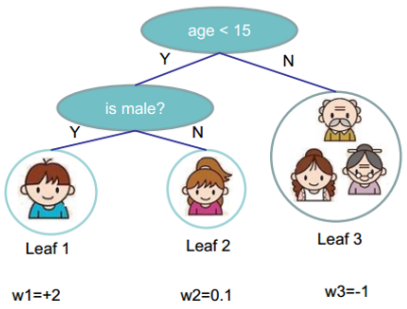
$$f_t(x) = w_{q(x)}, \quad w \in \mathbf{R}^T, q : \mathbf{R}^d \to \{1, 2, \cdots, T\}$$




具体表现示例如下，引入正则化项 $\Omega(f_t)$来控制树的复杂度，从而实现有效的控制模型的过拟合，**这是xgboost中的第一个重要点。式子中的$T$为叶子节点数**

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

Number of leaves　　L2 norm of leaf scores




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

### 5.2 XGBoost中的Boosting Tree模型


2018年7月(37)

2018年6月(3)

2018年2月(2)

最新评论

1. Re:文本分类实战（
M + Attention模型

博主您好，能问您一个
本序列是变长的，在双
加入seqence_length
以吗，为什么我加了之
也不报错，但是不会动。
了长度...

2. Re:文本分类实战（
M模型

还有，这个tf.nn.bidire
c_rnn返回的outputs中
向，前向没有问题，后
s，我输出了一下，在有
后向输出依然是（...

--

3. Re:文本分类实战（
M模型

博主，想问您一下，您
irectional_dynamic_
uence_length参数，
充的情况下不是会导致
有问题吗？...

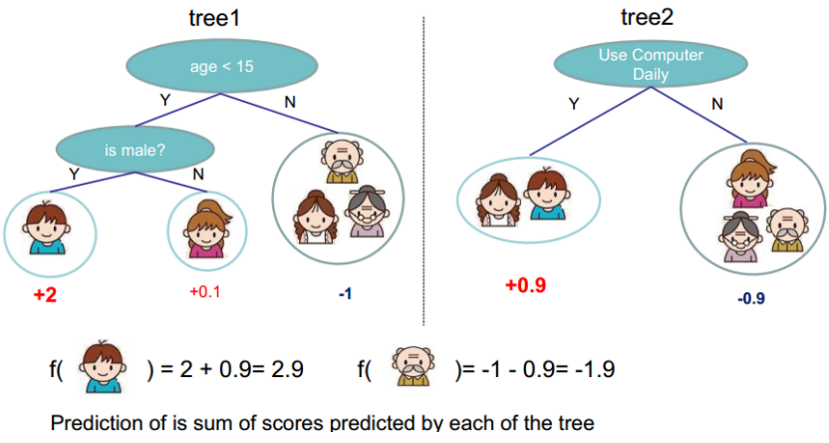
--

4. Re:文本分类实战（
M + Attention模型

Instructions for upda
on will only be availa
v1 compatibility libra

5. Re:文本分类实战（
N 模型

@ qind你可以自己根据
代，没有说一般迭代多




和GBDT方法一样，XGBoost的提升模型也是采用残差，不同的是分裂结点选取的时候不一定是最小平方损失，其损失函数如下，较GBDT其根据树模型的复杂度加入了一项正则化项：

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

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

### 5.3 对目标函数进行改写

- Goal $Obj^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + constant$
  - Seems still complicated except for the case of square loss
- Take Taylor expansion of the objective
  - Recall $f(x + \Delta x) \simeq f(x) + f'(x)\Delta x + \frac{1}{2}f''(x)\Delta x^2$
  - Define $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}), \quad h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$

$$Obj^{(t)} \simeq \sum_{i=1}^{n} \left[ l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) + constant$$

上面的式子是通过泰勒展开式将损失函数展开为具有二阶导的平方函数。

在GBDT中我们通过求损失函数的负梯度（一阶导数），利用负梯度替代残差来拟合树模型。在XGBoost中直接用泰勒展开式将损失函数展开成二项式函数（**前提是损失函数一阶、二阶都连续可导，而且在这里计算一阶导和二阶导时可以并行计算**），假设此时我们定义好了树的结构（在后面介绍，和GBDT中直接用残差拟合不同），假设我们的叶节点区域为：

$$I_j = \{i | q(x_i) = j\}$$

上面式子中$i$代表样本$i$，$j$代表叶子节点$j$。

则我们的目标优化函数可以转换成（因为$l(y_i, y_i^{t-1})$是个已经确定的常数，可以舍去）：

$$\begin{aligned} Obj^{(t)} &\simeq \sum_{i=1}^{n} \left[ g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \\ &= \sum_{i=1}^{n} \left[ g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2 \right] + \gamma T + \lambda \frac{1}{2} \sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T} \left[ (\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2 \right] + \gamma T \end{aligned}$$

上面式子把样本都合并到叶子节点中了。

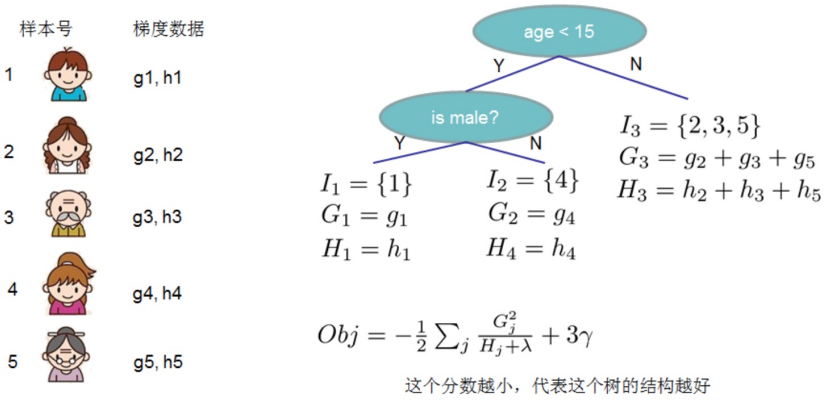
此时我们对$w_j$求导并令导数为0，可得：

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

其中 $G_j = \sum_{i \in I_j} g_i, H_j = \sum_{i \in T_j} h_j$。

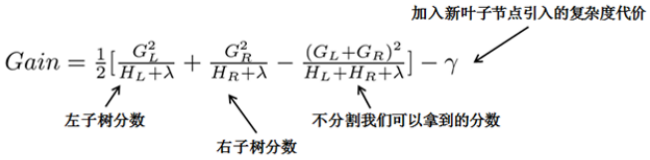
### 5.4 树结构的打分函数

上面的Obj值代表当指定一个树结构时，在目标上面最多减少多少，我们可以把它称为结构分数。可以认为这是一个类似与基尼指数一样更一般的对树结构进行打分的函数。如下面的例子所示

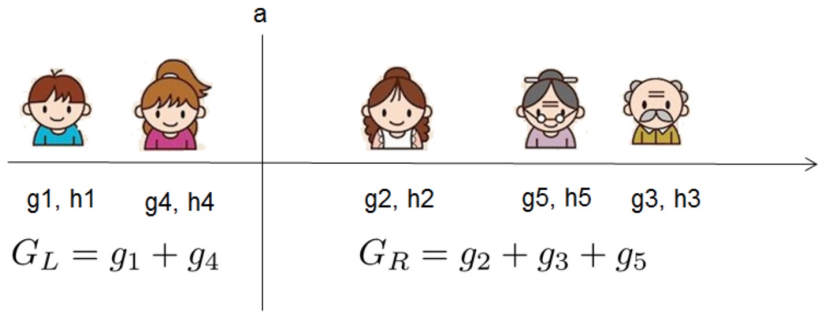



这个分数越小，代表这个树的结构越好

对于求得Obj分数最小的树结构，我们可以枚举所有的可能性，然后对比结构分数来获得最优的树结构，然而这种方法计算消耗太大，更常用的是贪心法（事实上绝大多数树模型都是这样的，只考虑当前节点的划分最优），每次尝试对已经存在的叶节点（最开始的叶节点是根节点）进行分割，然后获得分割后的增益为：




在这里以$Gain$作为判断是否分割的条件，这里的$Gain$可以看作是未分割前的Obj减去分割后的左右Obj，因此如果$Gain < 0$，则此叶节点不做分割，然而这样对于每次分割还是需要列出所有的分割方案（**对于特征的值的个数为n时，总共有2^n - 2 种划分**）。而实际中是采用近似贪心方法，我们先将所有样本按照$g_i$从小到大排序，然后进行遍历，查看每个节点是否需要分割（**对于特征的值的个数为$n$时，总共有$n-1$种划分**），具体示例如下：




最简单的树结构就是一个节点的树。我们可以算出这棵单节点的树的好坏程度obj*。假设我们现在想按照年龄将这棵单节点树进行分叉，我们需要知道：

1）按照年龄分是否有效，也就是是否减少了obj的值

2）如果可分，那么以哪个年龄值来分。

此时我们就是先将年龄特征从小到大排好序，然后再从左到右遍历分割

这样的分割方式，我们就只要对样本扫描一遍，就可以分割出$G_L$，$G_R$，然后根据$Gain$的分数进行分割，极大地节省了时间。所以从这里看，XGBoost中从新定义了一个划分属性，也就是这里的$Gain$，而这个划分属性的计算是由其目标损失决定obj的。

### 5.5 XGBoost中其他的正则化方法

1）像Adaboost和GBDT中一样，对每一个模型乘以一个系数$\lambda$（$0 < \lambda \leq 1$），用来降低每个模型对结果的贡献。

2）采用特征子采样方法，和RandomForest中的特征子采样一样，可以降低模型的方差

## 6、XGBoost和GBDT的区别

1）将树模型的复杂度加入到正则项中，来避免过拟合，因此泛化性能会由于GBDT。

2）损失函数是用泰勒展开式展开的，同时用到了一阶导和二阶导，可以加快优化速度。

3）和GBDT只支持CART作为基分类器之外，还支持线性分类器，在使用线性分类器的时候可以使用L1，L2正则化。

4）引进了特征子采样，像RandomForest那样，这种方法既能降低过拟合，还能减少计算。

5）在寻找最佳分割点时，考虑到传统的贪心算法效率较低，实现了一种近似贪心算法，用来加速和减小内存消耗，除此之外还考虑了稀疏数据集和缺失值的处理，对于特征的值有缺失的样本，XGBoost依然能自动找到其要分裂的方向。

6）XGBoost支持并行处理，XGBoost的并行不是在模型上的并行，而是在特征上的并行，将特征列排序后以block的形式存储在内存中，在后面的迭代中重复使用这个结构。这个block也使得并行化成为了可能，其次在进行节点分裂时，计算每个特征的增益，最终选

择增益最大的那个特征去做分割，那么各个特征的增益计算就可以开多线程进行。

分类： 机器学习


微笑sun
关注 - 18
粉丝 - 315
+加关注

2 0

« 上一篇： 机器学习算法总结(三)——集成学习(Adaboost、RandomForest)

» 下一篇： 机器学习算法总结(五)——聚类算法（K-means，密度聚类，层次聚类)

posted @ 2018-07-01 15:57 微笑sun 阅读(43916) 评论(2) 编辑 收藏

评论列表

#1楼 2019-01-06 00:46 BindyKeigo

你好 我想请问下 为什么obj化简为二次函数后 可以直接取最值 就保证最小化 难道二次项前面的系数Hj+λ必大于0吗？？

支持(0) 反对(0)

#2楼 [楼主] 2019-01-07 11:09 微笑sun

@ BindyKeigo
抱歉，这个我也不确定，不过在几个常见的二阶可导的损失函数上试过，hi的值确实是大于0的。以均方差和对数损失函数为例，他们的二阶导都是恒大于0.

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

**注册用户登录后才能发表评论，请 登录 或 注册， 访问 网站首页。**



**相关博文：**

· 随机森林，GBDT，XGBoost的对比
· 机器学习（四）--- 从gbdt到xgboost
· lightgbm,xgboost,gbdt的区别与联系
· GBDT为什么不能并行，XGBoost却可以
· GBDT,Adaboosting概念区分 GBDT与xgboost区别
» 更多推荐...

**最新 IT 新闻**:
· 微软Universal Print云端打印解决方案已转入私有预览
· 刘强东：“非典”时京东还是小公司，特别能体会疫情对企业的影响
· Gitee遭受DDoS攻击，官方建议不要在hosts里绑定IP地址
· 疫情下的口罩“疯狂制造”：物料飚涨40倍，3天回本
· Facebook在Microsoft Store上永久删除其桌面应用程序
» 更多新闻...