Gradient Boosting Decision Tree

读Greedy function approximation - A gradient boosting machine

Zheng Xin

目录

背景和相关工作

◆最速下降 Steepest Gradient Descent

➢ 核心思想：

无约束下，通过迭代的方式求函数$f(x)$的最优解，其中$f(x)$一阶可微。

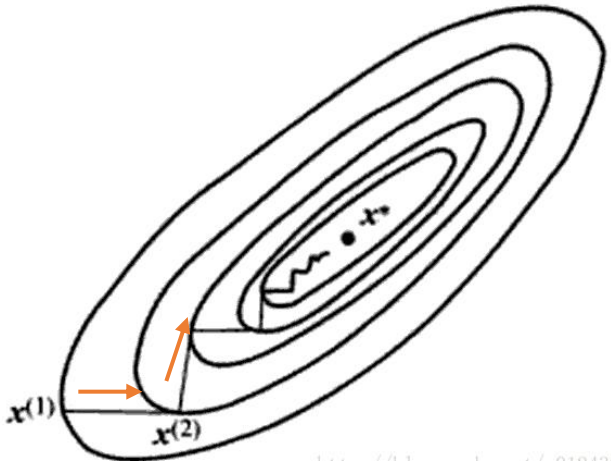
➢ 过程：

a) 给定一个初始点x_0；

b) 对i=1...k分别做如下迭代：

$x_i = x_{i-1} + pg_{i-1}$，g为f在x_{i-1}的负梯度，代表f在x_{i-1}处下降最快的方向。

c) 重复第二步，直到$|g|$或$|x_i - x_{i-1}|$足够小



➢以上迭代过程可以理解为：整个寻优的过程就向最陡的下坡路放小小跑的过程。

背景和相关工作

◆加性扩展 additive expansion

- 最速下降的寻优结果能可以表示成加和形式， m为迭代次数，即:

$$x^* = x_0 + p_1 x_1 + p_2 x_2 + \cdots + p_m x_m$$

- 在第k次迭代中，结果的更新规则是由前k-1次迭代的累计结果和第k-1次所在位置所决定。

$$x^k = x^{k-1} + p_k x_k$$

前k-1次迭代结果　　第k次迭代增量

可以将每次更新理解为，是对当前预测值和真实之之间差距的一次填充。

- Gradient Boosting （简称GB）正是由最速下降和加性扩展的思想启发而来。它也是一个寻优的过程，只不过上述过程寻找的是一个最优点，而GB寻找一个最优的函数。

主要思想

◆准备1:级联函数

➢给定$x=\{\mathrm{x}_1,\cdots,\mathrm{x}_n\}$和对应的输出$y$，进行学习预测。根据样本集$\{y_i,\mathrm{x}_i\}_1^N$估计（逼近）函数$\hat{F}(x)$。在样本集上得到最小化损失函数得到的函数，记为$\mathrm{F}^*(x)$。

$$F^*(x)=\arg\min_F E_{y,\mathbf{x}}L(y,F(\mathbf{x}))=\arg\min_F E_{\mathbf{x}}\left[E_y(L(y,F(x)))\,|x\right]$$

通常做法是将$F(x)$限制为带参函数族$F(x;P)$。其中$P=\{p_1,p_2,\cdots\}$，p_i选定以后就可以得到$F(x)$中的每个组成函数之后GBDT采用如下级联形式的公式：

$$F\left(\mathbf{x};\{\beta_m,\mathbf{a}_m\}_1^M\right)=\sum_{m=1}^{M}\beta_m h(\mathbf{x};\mathbf{a}_m)$$

➢上式中的通用式$h(x;a_{\mathrm{m}})$是关于x的带参函数，参数空间为$a=a_1,a_2,\cdots$。通过选定不同的a_m，得到不同的级联函数子项。

主要思想

◆准备2:带参函数在数值空间的优化

➢引入参数表达后，原目标优化函数式

$$F^*(x) = \arg\min_F E_{y,\mathbf{x}} L(y, F(\mathbf{x})) = \arg\min_F E_{\mathbf{x}} \left[E_y \left(L(y, F(x)) \right) | x \right]$$

改造为：

$$F^*(x) = F(x; \mathbf{P}^*) = \arg\min_p E_{y,\mathbf{x}} L(y, F(x; \mathbf{P}))$$

➢对于大多数$F(x;P)$和损失函数L，可以用数值优化算法进行求解。通常解的形式为：

$$\mathbf{P}^* = \sum_{m=0}^{M} \mathbf{p}_m$$

其中p_0随机初始化，$\{p_m\}_1^M$为随后每一步的增量，其值由优化算法决定。

主要思想

◆准备3:有限集上的SGD

➢在一个有限集合 $\{y_i, \mathrm{x}_i\}_1^N$ 估计(y, x)的联合分布，在这个集合上并不能得到确定的解$E_y[\cdot\,|x]$，还需要学习这些训练样本得到最优函数表达$\mathrm{F}^*(x)$，写成如下形式：

$$\{\beta_m, \mathbf{a}_m\}_1^M = \arg\min_{\{\beta'_m, a'_m\}_1^M} \sum_{i=1}^N L\left(y_i, F_{m-1}\left(\mathbf{x}_i\right) + \beta' h\left(\mathbf{x}_i; \mathbf{a}'\right)\right)$$

➢这个解可以尝试进行贪心的梯度下降方法，通过每一级的逼近，对于$m = 1,2, \ldots M$．每一级参数估计为：

$$\{\beta_m, \mathbf{a}_m\} = \arg\min_{\beta, \mathbf{a}} \sum_{i=1}^N L\left(y_i, F_{m-1}\left(\mathbf{x}_i\right) + \beta h\left(\mathbf{x}_i; \mathbf{a}\right)\right)$$

最后得到解函数的形式为：

$$F_m\left(\mathbf{x}\right) = F_{m-1}\left(\mathbf{x}_i\right) + \beta' h\left(\mathbf{x}_i; \mathbf{a}'\right)$$

➢在上述每一级的增加过程，逐级策略叫boosting，$h(x; a)$为弱分类器，通常为分类树，如果不引发歧义，可以理解为子函数。

主要思想

◆GB的一般套路

- 假设给定了前$m-1$级的逼近函数$\mathrm{F}_{m-1}(x)$，当前的子函数$\beta_m h(x;a)$可以视为最优解函数$F^*(x)$在梯度方向贪心策略的一步最佳逼近，子函数$h(x;a)$视为参数化函数类中最优的成员，每一个级联过程就等同于最速下降法的每一步，可以通过梯度得到子函数的参数估计：

$$-g_m(\mathbf{x}_i) = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}$$

- 我们可以得到第M步在训练集上关于$\mathrm{F}_{m-1}(x)$的最佳梯度$-\mathrm{g}_m = \{-\mathrm{g}_m(x)\}_1^N$。

- 但这些得到的梯度是从训练样本集中获取，并不能泛化到其他未知样本点。因此我们通过约束子函数$h(x;a)$，使它满足在训练样本上的值$h_m = \{h(x_i;a_m)\}_1^M$与$-g_m \in R^N$平行。

主要思想

- 根据上面的论述，子函数$h(x;a)$和$-g_m(x)$强相关，可以从下面式子中解出来：

$$\mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^{N} \left[-g_m\left(\mathbf{x}_i\right) - \beta h\left(\mathbf{x}_i;\mathbf{a}\right)\right]^2$$

- 用负梯度 $\{h(x_i;a_m)\}_1^M$替换$-g_m \in R^N$ 。

  有了梯度方向，我们还需要通过线搜索得到最佳步长：

$$\rho_m = \arg\min_{\rho} \sum_{i=1}^{N} L\left(y_i, F_{m-1}\left(\mathbf{x}_i\right) + \rho h\left(\mathbf{x}_i;\mathbf{a}_m\right)\right)$$

  那么第m级的函数逼近为：

$$F_m\left(\mathbf{x}\right) = F_{m-1}\left(\mathbf{x}\right) + \rho_m h\left(\mathbf{x};\mathbf{a}_m\right)$$

主要思想

◆GB的算法通用框架

1. $F_0(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^{N} L(y_i, \rho)$
2. $For\ m = 1\ to\ M\ do:$
3. $\tilde{y}_i = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}, i = 1, N$
4. $\mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^{N} \left[\tilde{y}_i - \beta h(\mathbf{x}_i; \mathbf{a})\right]^2$
5. $\rho_m = \arg\min_\rho \sum_{i=1}^{N} L(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i; \mathbf{a}_m))$
6. $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$
7. $endFor$

- 根据不同的损失函数，第4行的负梯度计算也不同，其他计算条件期望的拟合方法也可以用，例如最小绝对值误差(LAD)，Huber(M)。

主要思想

◆不同损失函数的GB

a) 最小均方差(LS)回归

损失函数为：

$$L(y,F)=\frac{(y-F)^2}{2}$$

在m轮对F_{m-1}求导得到梯度，负梯度为：

$$\tilde{y}_i=y_i-F_{m-1}(\mathbf{x}_i)$$

LS作为损失函数恰好与GB的残差的概念吻合，因此在通用框架中，第4行只需要简单地拟合当前的残差，第5行线搜索步长ρ_m可以等价为第4行中的β_m。因此LS_Boost流程如为：

$$\begin{aligned}
&For\ m=1\ to\ M\ do:\\
&\quad y_i=y_i-F_{m-1}(\mathbf{x}_i),\ i=1,N\\
&\quad (\rho_m,\mathbf{a}_m)=\arg\min_{\mathbf{a},\rho}\sum_{i=1}^{N}[\tilde{y}_i-\rho h(\mathbf{x}_i;\mathbf{a})]^2\\
&\quad F_m(\mathbf{x})=F_{m-1}(\mathbf{x})+\rho_m h(\mathbf{x};\mathbf{a}_m)\\
&endFor
\end{aligned}$$

主要思想

b) 最小绝对值误差(LAD)回归

损失函数为：

$$L(y, F) = |y - F|$$

可以得到负梯度为：

$$\tilde{y}_i = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})} = sign(y_i - F_{m-1}(\mathbf{x}_i))$$

此时需要一组分类器$h(x; a)$去拟合残差的符号，通过线搜索得到：

$$\begin{aligned}\rho_m &= \arg\min_{\rho} \sum_{i=1}^{N} |y_i - F_{m-1}(\mathbf{x}_i) - \rho h(\mathbf{x}_i; \mathbf{a}_m)| \\ &= \arg\min_{\rho} \sum_{i=1}^{N} |h(\mathbf{x}_i; \mathbf{a}_m)| \cdot \left|\frac{y_i - F_{m-1}(\mathbf{x}_i)}{h(\mathbf{x}_i; \mathbf{a}_m)} - \rho\right| \\ &= \text{median}_W \left\{ \frac{y_i - F_{m-1}(\mathbf{x}_i)}{h(\mathbf{x}_i; \mathbf{a}_m)} \right\}_1^N, w_i = |h(\mathbf{x}_i; \mathbf{a}_m)|.\end{aligned}$$

其中，$median_w\{\}$ 表示先进行w_i加权，在得到的结果中，取得中位数。

主要思想

c) Huber loss function

损失函数为：

$$L(y,F)=\begin{cases}\frac{1}{2}(y-F)^2, |y-F|<\delta\\ \delta\left(|y-F|-\frac{\delta}{2}\right), |y-F|>\delta\end{cases}$$

求负梯度：

$$\tilde{y}_i=-\left[\frac{\partial L\left(y_i,F\left(\mathbf{x}_i\right)\right)}{\partial F\left(\mathbf{x}_i\right)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}$$

$$=\begin{cases}y_i-F_{m-1}\left(\mathbf{x}_i\right), & \left|y_i-F_{m-1}\left(\mathbf{x}_i\right)\right|\leq\delta\\ \delta\cdot\operatorname{sign}\left(y_i-F_{m-1}\left(\mathbf{x}_i\right)\right), & \left|y_i-F_{m-1}\left(\mathbf{x}_i\right)\right|\leq\delta\end{cases}$$

根据负梯度，拟合$h(x;a)$：

$$\rho_m=\arg\min_{\rho}\sum_{i=1}^{N}L\left(y_i,F_{m-1}\left(\mathbf{x}_i\right)+\rho h\left(\mathbf{x}_i;\mathbf{a}_m\right)\right)$$

δ定义了残差里面哪部分属于异常值，这部分使用LAD约束，其他则使用LS损失函数约束。 δ由$y-\mathrm{F}^*(x)$决定，其中$\mathrm{F}^*(x)$由目标函数决定。实践中通常使用$|y-\mathrm{F}^*(x)|$分布的α-分位数的取值：

$$\delta_m=quantile_{\alpha}\{|y_i-F_{m-1}\left(\mathbf{x}_i\right)|\}_1^N$$

主要思想

得到第M轮的负梯度之后我们用一颗回归树进行拟合。在第j个节点R_{jm}：

$$\tilde{r}_{jm} = median_{\mathbf{x}_i \in R_{jm}} \{r_{m-1}(\mathbf{x}_i)\}$$

$$r_{m-1}(\mathbf{x}_i) = y_i - F_{m-1}(\mathbf{x}_i)$$

$$\gamma_{jm} = \tilde{r}_{jm} + \frac{1}{N_{jm}} \sum_{\mathbf{x}_i \in R_{jm}} \text{sign}\left(r_{m-1}(\mathbf{x}_i) - \tilde{r}_{jm}\right) \cdot \min\left(\delta_m, abs\left(r_{m-1}(\mathbf{x}_i) - \tilde{r}_{jm}\right)\right)$$

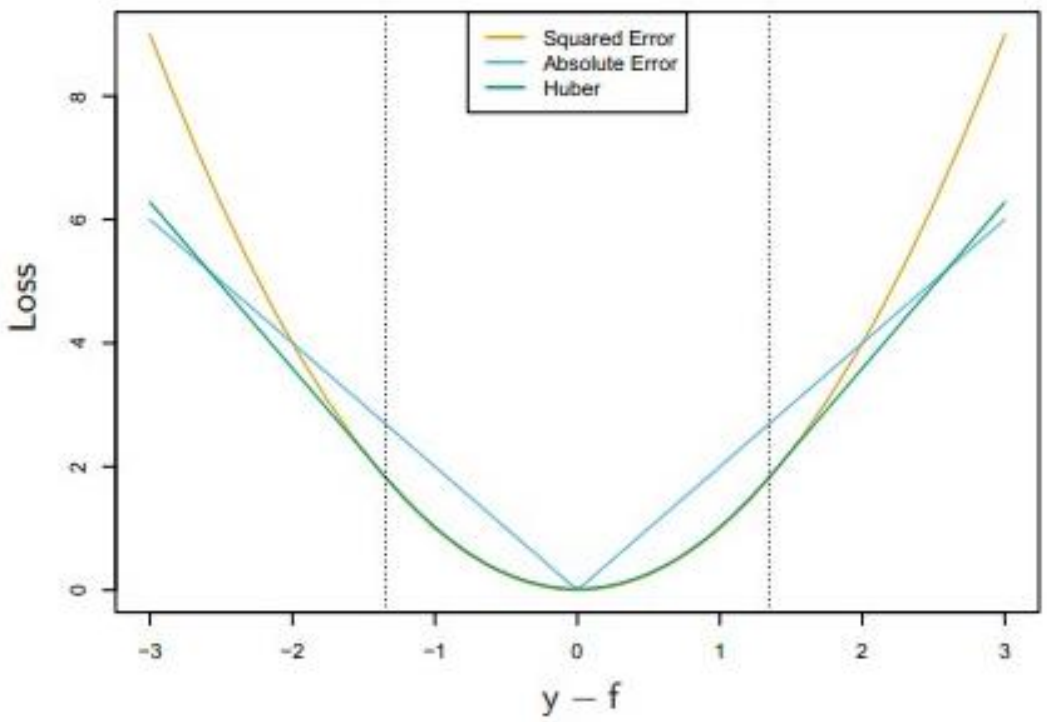
其中，N_{jm}为第j节点上观测值总数

因此M Boost的流程为：

$F_0(\mathbf{x}) = median\{y_i\}_1^N$
For $m = 1$ to M do:
$r_{m-1}(\mathbf{x}_i) = y_i - F_{m-1}(\mathbf{x}_i),\ i = 1, N$
$\delta_m = \text{quantile}_\alpha\{|r_{m-1}(\mathbf{x}_i)|\}_1^N$
$\tilde{y}_i = \begin{cases} r_{m-1}(\mathbf{x}_i), & |r_{m-1}(\mathbf{x}_i)| \le \delta_m \\ \delta_m \cdot \text{sign}(r_{m-1}(\mathbf{x}_i)), & |r_{m-1}(\mathbf{x}_i)| > \delta_m \end{cases},\ i = 1, N$
$\{R_{jm}\}_1^J = J\text{-terminal node } tree(\{\tilde{y}_i, \mathbf{x}_i\}_1^N)$
$\tilde{r}_{jm} = median_{\mathbf{x}_i \in R_{jm}}\{r_{m-1}(\mathbf{x}_i)\},\ j = 1, J$
$\gamma_{jm} = \tilde{r}_{jm} + \frac{1}{N_{jm}} \sum_{\mathbf{x}_i \in R_{jm}} sign\left(r_{m-1}(\mathbf{x}_i) - \tilde{r}_{jm}\right) \cdot \min(\delta_m, abs(r_{m-1}(\mathbf{x}_i) - \tilde{r}_{jm})),$
$j = 1, J$
$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \sum_{j=1}^J \gamma_{jm} 1(\mathbf{x} \in R_{jm})$
endFor
end Algorithm

主要思想

◆对比不同损失函数



- LAD只用了变量的顺序信息，负梯度只有{−1,1}二值，算法鲁棒性强。
- LS的定义与残差恰好吻合，可直接求得累加子函数时的参数值。
- 从图中可以看到以δ为分界点，huber损失函数中间呈现LS的性质，两边呈现LAD的线性。

主要思想

◆剪枝

- 通常，对训练集拟合太紧密会导致模型泛化能力的下降，正则化通过约束拟合过程能够减小过拟合的影响。
- 常用的技术手段是：

a) 通过控制决策树M的深度

在基函数选用决策树中，增加M值能够减小训练集的误差，但是太大会导致过拟合现象。最佳值M能够通过模型选择方法来估计，如使用独立测试集或交叉验证方法。

b) 缩减（Shrinkage）Gredient Boost。

缩减的方法是算法每次只学习一点点来减小单个特征对整体的影响，修改Gradient Boost的更新规则为：

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu \cdot \rho_m h(\mathbf{x}; \mathbf{a}_m), \qquad 0 < \nu \leq 1$$

主要思想

对比不同缩减参数v在不同迭代次数上的最小绝对误差：

ν	LS: $A(F_M(\mathbf{x}))$		LAD: $A(F_M(\mathbf{x}))$		L_2: $-2\log$ (like)		L_2: error rate	
1.0	15	0.48	19	0.57	20	0.60	436	0.111
0.5	43	0.40	19	0.44	80	0.50	371	0.106
0.25	77	0.34	84	0.38	310	0.46	967	0.099
0.125	146	0.32	307	0.35	570	0.45	580	0.098
0.06	326	0.32	509	0.35	1000	0.44	994	0.094
0.03	855	0.32	937	0.35	1000	0.45	979	0.097

可以看出，当v<0.125时达到稳定，表现较好。而迭代次数往往不稳定。

实例

为了在具体问题上对GBDT的过程理解，举一个例子。

如下表所示：一组数据，特征为年龄、体重，身高为标签值。共有5条数据，前四条为训练样本，最后一条为要预测的样本。

编号	年龄(岁)	体重（kg）	身高(m)(标签值)
1	5	20	1.1
2	7	30	1.3
3	21	70	1.7
4	30	60	1.8
5(要预测的)	25	65	?

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

1.初始化弱学习器:

由于此时只有根节点，样本 1，2， 3，4 都在根节点，此时要找到使得平方损失函数最小的参数YY，怎么求呢？平方损失显然是一个凸函数，直接求导，倒数等于零，得到Y。

$$\frac{\partial L(y_i, \Upsilon))}{\partial \Upsilon} = \frac{\partial(\frac{1}{2}[y-\Upsilon]^2)}{\partial \Upsilon} = \Upsilon - y$$

$$令 \Upsilon - y = 0，得 \Upsilon = y。$$

所以初始化时，YY取值为所有训练样本标签值的均值。

$$f_0(x) = \Upsilon = 1.475$$

实例

2.对迭代轮数m=1:

计算负梯度——残差

ri1=−[∂L(yi,f(xi)))∂f(xi)]f(x)=f0(x)ri1=−[∂L(yi,f(xi)))∂f(xi)]f(x)=f0(x)

说白了，就是残差（上面已经解释过了），在此例中，残差在下表列出：

编号	真实值	$f_0(x)$	残差
1	1.1	1.475	-0.375
2	1.3	1.475	-0.175
3	1.7	1.475	0.225
4	1.8	1.475	0.325

此时将残差作为样本的真实值训练f1(x)f1(x)，即下表数据

编号	年龄(岁)	体重（kg）	身高(m)(标签值)
1	5	20	-0.375
2	7	30	-0.175
3	21	70	0.225
4	30	60	0.325

实例

接着，寻找回归树的最佳划分节点，遍历每个特征的每个可能取值。从年龄特征的5开始，到体重特征的70结束，分别计算方差，找到使方差最小的那个划分节点即为最佳划分节点。

例如：以年龄7为划分节点，将小于7的样本划分为一类，大于等于7的样本划分为另一类。样本1为一组，样本2，3，4为一组，两组的方差分别为0，0.047，两组方差之和为0.047。所有可能划分情况如下表所示：

划分点	小于划分点的样本	大于等于划分点的样本	总方差
年龄5	/	1，2，3，4	0.082
年龄7	1	2，3，4	0.047
年龄21	1，2	3，4	0.0125
年龄30	1，2，3	4	0.062
体重20	/	1，2，3，4	0.082
体重30	1	2，3，4	0.047
体重60	1，2	3，4	0.0125
体重70	1，2，4	3	0.0867

以上划分点是的总方差最小为0.0125有两个划分点：年龄21和体重60，所以随机选一个作为划分点，这里我们选年龄21 。

实例

此时还需要做一件事情，给这两个叶子节点分别赋一个参数，来拟合残差。

$$\Upsilon_{j1} = \underbrace{arg\ min}_{\Upsilon} \sum_{x_i \in R_{j1}} L(y_i, f_0(x_i) + \Upsilon)$$

这里其实和上面初始化学习器是一个道理，平方损失，求导，令导数等于零，化简之后得到每个叶子节点的参数Y，其实就是标签值的均值。

根据上述划分节点：

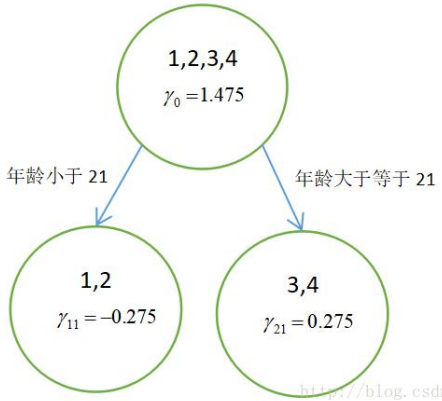
样本1，2为左叶子节点，(x1,x2∈R11)(x1,x2∈R11)，所以

Y11=(−0.375−0.175)/2=−0.275。

样本3，4为左叶子节点，(x3,x4∈R11)(x3,x4∈R11)，所以

Y21=(0.225+0.325)/2=0.275。

右图为第一次迭代所得的boost tree。



此时可更新强学习器 ：

$$f_1(x) = f_0(x) + \sum_{j=1}^{2} \Upsilon_{j1} I(x \in R_{j1})$$

实例

3.对迭代轮数m=2,3,4,5,…,M:

循环迭代M次，M是人为控制的参数，迭代结束生成M棵树。

4.得到最后的强学习器：

为了方别展示和理解，我们假设M = 1，根据上述结果得到强学习器：

$$f(x) = f_M(x) = f_0(x) + \sum_{m=1}^{M}\sum_{j=1}^{J} \Upsilon_{jm} I(x \in R_{jm})$$

$$= f_0(x) + \sum_{j=1}^{2} \Upsilon_{j1} I(x \in R_{j1})$$

5.预测样本5：

样本5在根节点中（即初始学习器）被预测为1.475，样本5的年龄为25，大于划分节点21岁，所以被分到了右边的叶子节点，同时被预测为0.275。此时便得到样本5的最总预测值为1.75

实验

◆实验一

- 数据集：Garne tdata(scientific)：岩石样本的化学浓度测量。N=13317
- 训练集：测试集=2:1
- 损失函数：最小绝对函数ν = 0.1
- 结

Terminal nodes	LS	LAD	*M*
2	0.58	0.57	0.57
3	0.48	0.47	0.46
4	0.49	0.45	0.45
6	0.48	0.44	0.43
11	0.47	0.44	0.43
21	0.46	0.43	0.43

从结果可看出，6个终端节点树被认为是足够的，并且仅使用三个终端节点树可以提供10%的最佳精度。

LADTreeBoost和MTreeBoostare的错误小于LSTreeBoost并且彼此相似。

实验

实验二

- 数据集Demographic data (demographic)： 商场中的社会问卷，选取的问卷数据如下表。 N=9409 。

Variable	Demographic	Number values	Type
1	sex	2	cat
2	martial status	5	cat
3	age	7	real
4	education	6	real
5	occupation	9	cat
6	income	9	real
7	years in Bay Area	5	real
8	dual incomes	2	cat
9	number in household	9	real
10	number in household<18	9	real
11	householder status	3	cat
12	type of home	5	cat
13	ethnic classification	8	cat
14	language in home	3	cat

- 本数据集有很多缺失数据，实验选用其他13个模型对收入建模，损失函数为最小绝对函数。
- 结果如下表：

Terminal nodes	LS	LAD	*M*
2	0.60	0.63	0.61
3	0.60	0.62	0.59
4	0.59	0.59	0.59
6	0.59	0.58	0.59
11	0.59	0.57	0.58
21	0.59	0.58	0.58

可以从表中看出，三种方法的性能差异不大。
由于数据的高度分散，没有外延或长尾分布。
当组成树大小J增加时，误差也几乎没有减少，这表明输入变量之间缺乏相互作用;单个输入变量（J = 2）中的近似加法似乎是足够的。

结论

- GBDT的核心就在于，每一棵树学的是之前所有树结论和的残差，这个残差就是一个加预测值后能得真实值的累加量。

- 使用不同的损失函数，对模型的影响效果不同。

- 为防止过度拟合，可以采用控制决策树深度和设置缩减参数的方法。

- 本论文详细阐述了Gradient Boosting的框架，并对实践过程中具体损失函数、剪枝、影响模型效果的因素，以及在实际数据集的结果，进行了详细分析。本ppt重点对GB模型的产生和程序运行过程进行了分析，主要目的是使初接触者明白其中的产生原理，以及在具体问题上有一个具象的理解，但略过了影响模型效果的论文内容。

谢谢!

2019-3-3