1. AdaBoost

1.1. AdaBoost算法

- 输入：训练集$T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}, x_i\in\mathcal{X}\subseteq\mathbf{R^n},\mathbf{y_i}\in\{\mathbf{-1},\mathbf{+1}\}$
- 输出：最终分类器$G(x)$
- 初始化训练数据的分布$D_1=(w_{11},\cdots,w_{1i},\cdots,w_{1N}), w_{1i}=\frac{1}{N}$
- 对于$m=1,2,\cdots,M$
  - 基本分类器 $G_m(x):\mathcal{X}\rightarrow\{-1,+1\}$
  - 计算$G_m(x)$在训练数据集上的分类误差率：$e_m=\sum_{i=1}^N P\{G_m(x_i)\neq y_i\}=\sum_{i=1}^N w_{mi}I\{G_m(x_i)\neq y_i\}$
  - 计算$G_m(x)$的系数：$\alpha_m=\frac{1}{2}\log\frac{1-e_m}{e_m}$
  - 更新训练数据集的权值分布：

$$
\begin{aligned}
D_{m+1} &= (w_{m+1,1},\cdots,w_{m+1,i},\cdots,w_{m+1,N}) \\
w_{m+1,i} &= \frac{w_{mi}}{Z_m}\exp\{-\alpha_m y_i G_m(x_i)\} = \begin{cases} \frac{w_{mi}}{Z_m}e^{-\alpha_m} & \text{if } G_m(x_i)=y_i \\ \frac{w_{mi}}{Z_m}e^{\alpha_m} & \text{if } G_m(x_i)\neq y_i \end{cases} \\
Z_m &= \sum_{i=1}^N w_{mi}\exp\{-\alpha_m y_i G_m(x_i)\}
\end{aligned}
$$

- 构建基本分类器的线性组合：$f(x)=\sum_{m=1}^M \alpha_m G_m(x)$
- 得到最终分类器：$G(x)=\text{sign}(f(x))=\text{sign}\{\sum_{m=1}^M \alpha_m G_m(x)\}$

当$e_m \leq \frac{1}{2}$时，$\alpha_m \geq 0$，且α_m随着e_m的减小而增大，所以分类误差率越小的基本分类器在最终分类器中的作用越大。

1.2. 前向分布算法

AdaBoost算法另一个解释，可认为AdaBoost算法是模型为加法模型、损失函数为指数函数、学习算法为前向分布算法时的二分类学习方法。

- 加法模型：$f(x)=\sum_{m=1}^{M}\beta_m b(x;\gamma_m)$
- $b(x;\gamma_m)$为基函数
- γ_m为基函数的参数
- β_m为基函数的系数
- 在给定训练数据及损失函数$L(y,f(x))$的条件下，学习加法模型$f(x)$成为**经验风险极小化**即**损失函数极小化**问题：$\min_{\beta_m,\gamma_m}\sum_{i=1}^{N}L\left(y_i,\sum_{m=1}^{M}\beta_m b(x_i;\gamma_m)\right)$

前向分布算法求解这一损失函数优化问题的想法是：从前向后，每一步只学习一个基函数及其系数，逐步逼近优化目标函数。具体地，每步只需优化损失函数$\min_{\beta,\gamma}\sum_{i=1}^{N}L\left(y_i,\beta b(x_i;\gamma)\right)$

前向分布算法：

- 输入：训练集$T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\},x_i\in\mathcal{X}\subseteq\mathbf{R^n},\mathbf{y_i}\in\{\mathbf{-1},\mathbf{+1}\}$；损失函数$L(y,f(x))$；基函数集$\{b(x;\gamma)\}$
- 输出：加法模型$f(x)$
- 初始化$f_0(x)=0$
- 对于$m=1,2,\cdots,M$
  - 极小化损失函数$(\beta_m,\gamma_m)=\arg\min_{\beta,\gamma}\sum_{i=1}^{N}L\left(y_i,f_{m-1}(x_i)+\beta b(x_i;\gamma)\right)$得到参数$\beta_m,\gamma_m$
  - 更新$f_m(x)=f_{m-1}(x)+\beta_m b(x;\gamma_m)$

前向分布算法将同时求解从$m=1$到M所有参数β_m,γ_m的优化问题简化为逐次求解各个β_m,γ_m的优化问题。

1.3. 前向分布算法与AdaBoost

定理：AdaBoost算法是前向分布加法算法的特例。这时，模型是由基本分类器组成的加法模型，损失函数是指数函数。

证明前向分布算法的损失函数是指数损失函数$L\left(y,f(x)\right)=\exp\left(-yf(x)\right)$时，其学习的具体操作等价于AdaBoost算法学习的具体操作。

- 假设经过$m-1$轮迭代前向分布算法，已经得到$f_{m-1}(x)=f_{m-2}(x)+\alpha_{m-1}G_{m-1}(x)=\alpha_1G_1(x)+\cdots+\alpha_{m-1}G_{m-1}(x)$.
- 在第m轮迭代得到$\alpha_m,G_m(x),f_m(x)$
- 目标是使前向分布算法得到的$\alpha_m,G_m(x)$使$f_m(x)$在训练数据集T上的指数损失最小，即

$$
\begin{aligned}
\left(\alpha_m, G_m(x)\right) &= \arg\min_{\alpha,G} \sum_{i=1}^{N} \exp\left[-y_i\left(f_{m-1}(x_i)+\alpha G(x_i)\right)\right] \\
&= \arg\min_{\alpha,G} \sum_{i=1}^{N} \bar{w}_{mi} \exp[-y_i \alpha G(x_i)] \\
\bar{w}_{mi} &= \exp[-y_i f_{m-1}(x_i)]
\end{aligned} \tag{1}
$$

$\bar{w}_{mi}$既不依赖α也不依赖于G，所以与最小化无关。
$\bar{w}_{mi}$依赖于$f_{m-1}(x)$，随着每一轮迭代而发生改变。

证明使式(1)达到最小的$\alpha_m^*, G_m^*(x)$就是AdaBoost算法所得到的$\alpha_m, G_m(x)$

- step1 求解$G_m^*(x)$
  对任意$\alpha > 0$，使式(1)最小的$G(x)$由下式得到：

$$
\begin{aligned}
G_m^*(x) &= \arg\min_{G} \sum_{i=1}^{N} \bar{w}_{mi} I\left(y_i \neq G(x_i)\right) \\
\bar{w}_{mi} &= \exp[-y_i f_{m-1}(x_i)]
\end{aligned}
$$

- step2 求解α_m^*

$$
\sum_{i=1}^{N} \bar{w}_{mi} \exp[-y_i \alpha G(x_i)] = \sum_{y_i = G_m(x_i)} \bar{w}_{mi} e^{-\alpha} + \sum_{y_i \notin G_m(x_i)} \bar{w}_{mi} e^{\alpha}
$$

2. 决策树

- 设训练数据集为D，$|D|$表示其样本容量，即样本个数
- 设有K个类$C_k, k=1,2,\cdots,K$，$|C_k|$为属于类C_k的样本个数，$\sum_{k=1}^{K}|C_k| = |D|$
- 设特征A由n个不同的取值$\{a_1, a_2, \cdots, a_n\}$，根据特征$A$的取值将$D$划分为$n$个子集$D_1, D_2, \cdots, D_n$，$|D_i|$为$D_i$的样本个数，$\sum_{i=1}^{n}|D_i| = |D|$
- 记子集D_i中属于类C_k的样本集合为D_{ik}，即$D_{ik} = D_i \cap C_k$，$|D_{ik}|$为D_{ik}的样本个数
- 数据集D的经验熵$H(D) = -\sum_{k=1}^{K}\frac{|C_k|}{|D|}\log_2\frac{|C_k|}{|D|}$
- 特征A对数据集D的经验条件熵$H(D|A) = \sum_{i=1}^{n}\frac{|D_i|}{|D|}H(D_i) = -\sum_{i=1}^{n}\frac{|D_i|}{|D|}\sum_{k=1}^{K}\frac{|D_{ik}|}{|D_i|}\log_2\frac{|D_{ik}|}{|D_i|}$
- 信息增益$g(D,A) = H(D) - H(D|A)$
- 特征A对数据集D的信息增益比$g_R(D,A) = \frac{g(D,A)}{H_A(D)}, H_A(D) = -\sum_{i=1}^{n}\frac{|D_i|}{|D|}\log_2\frac{|D_i|}{|D|}$，$n$是特征$A$取值的个数

A_1年龄	类别	数量
青年	是	2
青年	否	3
中年	是	3

A_1年龄	类别	数量
中年	否	2
老年	是	4
老年	否	1

$$
\begin{aligned}
H(D) &= -\frac{9}{15}\log_2\frac{9}{15}-\frac{6}{15}\log_2\frac{6}{15}=0.971 \\
g(D,A_1) &= H(D)-[\frac{5}{15}H(D_1)+\frac{5}{15}H(D_2)+\frac{5}{15}H(D_3)] \\
&= 0.971-[\frac{5}{15}(-\frac{2}{5}\log_2\frac{2}{5}-\frac{3}{5}\log_2\frac{3}{5})+\frac{5}{15}(-\frac{3}{5}\log_2\frac{3}{5}-\frac{2}{5}\log_2\frac{2}{5})+\frac{5}{15}(-\frac{4}{5}\log_2\frac{4}{5}-\frac{1}{5}\log_2\frac{1}{5})] \\
&= 0.971-0.888=0.083
\end{aligned}
$$

A_2有工作	类别	数量
是	是	5
是	否	0
否	是	4
否	否	6

$$
\begin{aligned}
g(D,A_2) &= H(D)-[\frac{5}{15}H(D_1)+\frac{10}{15}H(D_2)] \\
&= 0.971-[\frac{5}{15}(-\frac{5}{5}\log_2\frac{5}{5}-\frac{0}{5}\log_2\frac{0}{5})+\frac{10}{15}(-\frac{4}{10}\log_2\frac{4}{10}-\frac{6}{10}\log_2\frac{6}{10})] \\
&= 0.324
\end{aligned}
$$

A_3有自己的房子	类别	数量
是	是	6
是	否	0
否	是	3
否	否	6

$$
\begin{aligned}
g(D,A_3) &= H(D)-[\frac{6}{15}H(D_1)+\frac{9}{15}H(D_2)] \\
&= 0.971-[\frac{6}{15}(-\frac{6}{6}\log_2\frac{6}{6}-\frac{0}{6}\log_2\frac{0}{6})+\frac{9}{15}(-\frac{3}{9}\log_2\frac{3}{9}-\frac{6}{9}\log_2\frac{6}{9})] \\
&= 0.420
\end{aligned}
$$

A_4信贷情况	类别	数量
一般	是	1
一般	否	4

A_4信贷情况	类别	数量
好	是	4
好	否	2
非常好	是	4
非常好	否	0

$$\begin{aligned} g(D, A_4) &= H(D) - [\frac{5}{15}H(D_1) + \frac{6}{15}H(D_2) + \frac{4}{15}H(D_3)] \\ &= 0.971 - [\frac{5}{15}(-\frac{1}{5}\log_2\frac{1}{5} - \frac{4}{5}\log_2\frac{4}{5}) + \frac{6}{15}(-\frac{4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6}) + \frac{4}{15}(-\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4})] \\ &= 0.971 - 0.608 = 0.363 \end{aligned}$$

由于特征A_3（有自己的房子）的信息增益值最大，所以选择特征A_3作为最优特征。

2.1. ID3与C4.5

2.1.1. 算法

ID3算法:

- 输入：训练数据集D，特征集A，阈值ϵ
- 输出：决策树T
- （1）若D中所有实例属于同一类C_k，则T为单结点树，并将类C_k作为该结点的类标记，返回T；
- （2）若$A=\varnothing$，则T为单结点树，并将D中实例数最大的类C_k作为该结点的类标记，返回T
- （3）否则，计算A中各特征对D的信息增益，选择信息增益最大的特征A_g
- （4）若A_g的信息增益小于阈值ϵ，则置T为单结点树，并将D中实例数最大的类C_k作为该结点的类标记，返回T
- （5）否则，对A_g的每一可能值a_i，依$A_g=a_i$将D分割为若干非空子集D_i，将D_i中实例数最大的类作为标记，构建子结点，由结点及其子结点构成树T，返回T
- （6）对于第i个子结点，以D_i为训练集，以$A-\{A_g\}$为特征集，递归地调用步（1）～（5），得到子树T_i，返回T_i

C4.5算法： 将ID3中的「信息增益」改为「信息增益比」来选择特征。

2.1.2. 决策树剪枝

- 「决策树生成」只考虑了通过提高信息增益（或信息增益比）对训练数据进行更好的拟合
- 「决策树剪枝」通过优化损失函数，还考虑了减小模型复杂度
- 「决策树生成」学习局部的模型
- 「决策树剪枝」学习整体的模型

设树T的叶结点个数为$|T|$，t是树T的叶结点，该叶结点有N_t个样本点，其中k类的样本点有$N_{tk}, k=1,2,\cdots,K$个，$H_t(T)$为叶结点t上经验熵，$\alpha \geq 0$为参数，则决策树学习的损失函数可以定义为$C_\alpha = \sum_{t=1}^{|T|} N_t H_t(T) + \alpha|T|$，其中经验熵为$H_t(T) = -\sum_k \frac{N_{tk}}{N_t}\log\frac{N_{tk}}{N_t}$

- $C(T) = \sum_{t=1}^{|T|} N_t H_t(T) = -\sum_{t=1}^{|T|}\sum_{k=1}^{K} N_{tk}\log\frac{N_{tk}}{N_t}$
- $C_\alpha(T) = C(T) + \alpha|T|$

- $C(T)$表示模型对训练数据的预测误差，即模型与训练数据的拟合程度
- $|T|$表示模型复杂度，参数$\alpha \geq 0$控制两者之间的影响
- 较大的α促使选择较简单的模型（树），较小的α促使选择较复杂的模型（树）
- 剪枝，就是当α确定时，选择损失函数最小的模型，即损失函数最小的子树
- 当α确定时，子树越大，往往与训练数据拟合的越好，但是模型的复杂度就越高；相反，子树越小，模型的复杂度就越低，但是往往与训练数据的拟合不好

树的剪枝算法：

- 输入：生成算法产生的整个树T，参数α
- 输出：修剪后的子树T_α
- （1）计算每个结点的经验熵
- （2）递归地从树的叶节点向上回缩
- （3）设一组叶节点回缩到其父结点之前与之后的整体树分别为T_B与T_A，其对应的损失函数值分别是$C_\alpha(T_B)$与$C_\alpha(T_A)$，若$C_\alpha(T_A) \leq C_\alpha(T_B)$，则进行剪枝，即将父结点变为新的叶结点
- 返回（2），直至不能继续为止，得到损失函数最小的子树T_α

2.2. CART

3. GBDT

3.1. Algorithm1 Gradient_Boost

$$
\begin{aligned}
&F_0(x) = \arg\min_\rho \sum_{i=1}^{N} L(y_i, \rho) \\
&\text{For } m = 1 \text{ to } M \text{ do:} \\
&\tilde{y}_i = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}, i = 1, N \\
&\alpha_m = \arg\min_{\alpha,\beta} \sum_{i=1}^{N} [\tilde{y}_i - \beta h(x_i; \alpha)]^2 \\
&\rho_m = \arg\min_\rho \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \rho h(x_i; \alpha_m)) \\
&F_m(x) = F_{m-1}(x) + \rho_m h(x; \alpha_m) \\
&\text{endFor} \\
&\text{end Algorithm}
\end{aligned}
$$

函数$h(x;\alpha)$是weaker learner或base learner，通常是分类树。

3.2. Algorithm2 LeastSquares_Boost

$$L(y,F) = \frac{1}{2}(y-F)^2$$
$$F_0(x) = \bar{y}$$
For $m = 1$ to M do:
$$\tilde{y}_i = y_i - F_{m-1}(x_i)$$
$$(\rho_m, \alpha_m) = \arg\min_{\alpha,\rho} \sum_{i=1}^{N} [\tilde{y}_i - \rho h(x_i;\alpha)]^2$$
$$F_m(x) = F_{m-1}(x) + \rho_m h(x;\alpha_m)$$
endFor
end Algorithm

3.3. Algorithm3 LeastAbsoluteDeviation_TreeBoost

$$L(y,F) = |y-F|$$
$$\tilde{y}_i = \text{sign}\{y_i - F_{m-1}(x_i)\} = -[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}]_{F(x)=F_{m-1}(x)}$$
$$F_0(x) = \text{median}\{y_i\}_1^N$$
For $m = 1$ to M do:
$$\tilde{y}_i = \text{sign}\{y_i - F_{m-1}(x_i)\}, i = 1, N$$
$$\{R_{jm}\}_1^J = \text{J-terminal node tree}(\{\tilde{y}_i, x_i\}_1^N)$$
$$\gamma_{jm} = \text{median}_{x_i \in R_{jm}} \{y_i - F_{m-1}(x_i)\}, j = 1, J$$
$$F_m(x) = F_{m-1}(x) + \sum_{j=1}^{J} \gamma_{jm} 1(x \in R_{jm})$$
endFor
end Algorithm

3.4. Algorithm4 M_TreeBoost

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

$F_0(x)=\text{median}\{y_i\}_1^N$

For $m=1$ to M do:

$r_{m-1}(x_i)=y_i-F_{m-1}(x_i), i=1,N$

$\delta_m=\text{quantile}_\alpha\{|r_{m-1}(x_i)|\}_1^N$

$$\widetilde{y}_i=\begin{cases}r_{m-1}(x_i), |r_{m-1}(x_i)|\le\delta_m\\ \delta_m\cdot\text{sign}(r_{m-1}(x_i)), |r_{m-1}(x_i)|>\delta_m\end{cases}$$

$\{R_{jm}\}_1^J=\text{J-terminal node tree}(\{\widetilde{y}_i,x_i\}_1^N)$

$\widetilde{r}_{jm}=\text{median}_{x_i\in R_{jm}}\{r_{m-1}(x_i)\}, j=1,J$

$$\gamma_{jm}=\widetilde{r}_{jm}+\frac{1}{N_{jm}}\sum_{x_i\in R_{jm}}\text{sign}(r_{m-1}(x_i)-\widetilde{r}_{jm})\cdot\min(\delta_m,\text{abs}(r_{m-1}(x_i)-\widetilde{r}_{jm})), j=1,J$$

$$F_m(x)=F_{m-1}(x)+\sum_{j=1}^J\gamma_{jm}1(x\in R_{jm})$$

endFor

end Algorithm

3.5. Algorithm5 L_K_TreeBoost(two-class logistic)

$$L(y,F)=\log(1+\exp(-2yF)), y\in\{-1,1\}$$
$$F(x)=\frac{1}{2}\log[\frac{\Pr(y=1|x)}{\Pr(y=-1|x)}]$$
$$F_0(x)=\frac{1}{2}\log\frac{1+\tilde{y}}{1-y}$$
For $m=1$ to M do:
$$\tilde{y}_i=\frac{2y_i}{1+\exp(2y_iF_{m-1}(x_i))}, i=1,N$$
$$\{R_{jm}\}_1^J=\text{J-terminal node tree}(\{\tilde{y}_i,x_i\}_1^N)$$
$$\gamma_{jm}=\sum_{x_i\in R_{jm}}\tilde{y}_i/\sum_{x_i\in R_{jm}}|\tilde{y}_i|(2-|\tilde{y}_i|), j=1,J$$
$$F_m(x)=F_{m-1}(x)+\sum_{j=1}^J\gamma_{jm}1(x\in R_{jm})$$
$$p_+(x)=\hat{\Pr}(y=1|x)=\frac{1}{1+e^{-2F_M(x)}}$$
$$p_-(x)=\hat{\Pr}(y=-1|x)=\frac{1}{1+e^{2F_M(x)}}$$
$$\hat{y}(x)=2\cdot 1[c(-1,1)p_+(x)>c(1,-1)p_-(x)]-1$$
$c(\hat{y},y)$ is the cost associated with predicting $\hat{y}$ when the truth is y

$$-2F_M(x)=\log[\frac{\Pr(y=-1|x)}{\Pr(y=1|x)}]$$
$$e^{-2F_M(x)}=\frac{\Pr(y=-1|x)}{\Pr(y=1|x)}=\frac{p}{1-p}$$
$$1+e^{-2F_M(x)}=\frac{1-p+p}{1-p}=\frac{1}{1-p}$$

对于二分类的logistic回归问题，在第m次迭代的经验损失函数为$\phi(\rho,\alpha)=\sum_{i=1}^N\log[1+\exp(-2y_iF_{m-1}(x_i))\cdot\exp(-2y_i\rho h(x_i;\alpha))]$。
如果$y_iF_{m-1}(x_i)$非常大，则经验损失函数值接近于0，即$(\rho_m,\alpha_m)=\arg\min_{\rho,\alpha}\phi_m(\rho,\alpha)$，这表明对于计算$y_iF_{m-1}(x_i)$非常大的观察值可以从第$m$次迭代中删除。因此，$w_i=\exp(-2y_iF_{m-1}(x_i))$可被视为衡量基于估计$\rho_mh(x;\alpha_m)$的第$i$次观测的影响或权重的测量方法。
另一种衡量在第m次迭代中基于估计$\rho_mh(x;\alpha_m)$的第i次观测的影响或权重：$w_i=|\tilde{y}_i|(1-|\tilde{y}_i|)$，influence trimming删除$w_i<w_{l(\alpha)}$的所有观测值，其中$\sum_{i=1}^{l(\alpha)}w_{(i)}=\alpha\sum_{i=1}^N w_i$，$\{w_{(i)}\}_1^N$是以升序排序的权重，$\alpha\in[0.05,0.2]$。

3.6. Algorithm6 L_K_TreeBoost(multi-class logistic)

$$L(\{y_k, F_k(x)\}_1^K) = -\sum_{k=1}^{K} y_k \log p_k(x), y_k = 1(class = k) \in (0,1), p_k = Pr(y_k = 1|x)$$

$$F_k(x) = \log p_k(x) - \frac{1}{K}\sum_{l=1}^{K} \log p_l(x)$$

$F_{k0} = 0, k = 1, K$

For $m = 1$ to M do:

$$p_k(x) = \exp(F_k(x)) / \sum_{l=1}^{K} \exp(F_l(x)), k = 1, K$$

For $k = 1$ to K do:

$\widetilde{y}_{ik} = y_{ik} - p_k(x_i), i = 1, N$

$\{R_{jkm}\}_{j=1}^J =$ J-terminal node tree$(\{\widetilde{y}_{ik}, x_i\}_1^N)$

$$\gamma_{jkm} = \frac{K-1}{K}\frac{\sum_{x_i \in R_{jkm}} \widetilde{y}_{ik}}{\sum_{x_i \in R_{jkm}} |\widetilde{y}_{ik}|(1 - |\widetilde{y}_{ik}|)}, j = 1, J$$

$$F_{km}(x) = F_{k,m-1}(x) + \sum_{j=1}^{J} \gamma_{jkm} 1(x \in R_{jkm})$$

endFor

endFor

end Algorithm

$$\hat{k}(x) = \arg\min_{1 \le k \le K} \sum_{k'=1}^{K} c(k, k^{'}) p_{k'M}(x)$$

3.7. 回归树

考虑每个基学习器都是一个有J个叶节点的回归树，每个回归树有自己additive形式$h(x; \{b_j, R_j\}_1^J) = \sum_{j=1}^J b_j 1(x \in R_j)$，此处$\{R_j\}_1^J$是预测变量$x$的所有可能取值的不相连的区域。这些区域由相关树的叶节点来表示。当条件正确时，指示函数$1(\cdot)$为1，否则为0。
由于区域是不相连的，$h(x; \{b_j, R_j\}_1^J) = \sum_{j=1}^J b_j 1(x \in R_j)$等价于预测规则：如果$x \in R_j$，则$h(x) = b_j$。

$F_m(x) = F_{m-1}(x) + \rho_m h(x; \alpha_m) \Rightarrow F_m(x) = F_{m-1}(x) + \rho_m \sum_{j=1}^J b_{jm} 1(x \in R_{jm})$，其中$\{R_{jm}\}_1^J$是第$m$次迭代树的叶节点所定义的区域。$b_{jm} = \text{ave}_{x_i \in R_{jm}} \widetilde{y}_i$

4. XGBoost

5. LightGBM

6. CatBoost