Boosting

迭代过程

机器学习 -> 统计学习方法 -> 补充

理论基础

名词解释

集成学习：通过构建并结合多个学习器来完成学习任务。

同质 homogeneous：决策树集成中全是决策树，神经网络集成中全是神经网络。

基学习器 base learner：同质集成中的个体学习器。

基学习算法 base learning algorithm：基学习器所使用的学习算法。

异质 heterogenous：集成包含不同类型的个体学习器。

组件学习器 component learner：和基学习器对应，它们统称为个体学习器。

集成学习的可行性证明

假设二分类问题 $y \in \{-1, +1\}$ 和真实函数 f，假定基分类器的错误率是 ϵ，即对每个**基分类器** h_i 有：

$$P\left(h_i(\boldsymbol{x}) \neq f(\boldsymbol{x})\right) = \epsilon$$

假设集成通过投票结合 T 个基分类器，若有超过半数的基分类器正确，则集成分类就正确：

$$H(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{i=1}^{T} h_i(\boldsymbol{x})\right)$$

根据 Hoeffding 不等式，得到集成后的错误率：

$$\begin{aligned} P(H(\boldsymbol{x}) \neq f(\boldsymbol{x})) &= \sum_{k=0}^{\lfloor T/2 \rfloor} \binom{T}{k} (1-\epsilon)^k \epsilon^{T-k} \\ &\leqslant \exp\left(-\frac{1}{2} T (1-2\epsilon)^2\right) \end{aligned}$$

> $P(H(n) \leqslant k)$ 是另一种写法，含义相同。
>
> 由这条表达式，我们有：
>
> $$h_i(x) = \begin{cases} 1 & C_n^x p^x (1-p)^{n-x} >= 0.5 \\ -1 & C_n^x p^x (1-p)^{n-x} < 0.5 \end{cases}$$

第一个等号表示 n 个基学习器中分类正确的个数小于 k 的概率。若假定集成通过简单投票法结合 n 个分类器，超过半数的基学习器正确，则集成分类就正确，即临界值 $k = 0.5 * n = (1 - \epsilon - \delta)n$ 。

第二个等号的 Hoeffding 不等式的定义，$\delta > 0$：

$$P(H(n) \leqslant (p-\delta)n) \leqslant e^{-2\delta^2 n}$$

其中 $\binom{T}{k}$ 表示 C_T^k，$\delta = 0.5 - \epsilon$ 。

> Ps: n 和 T 等价。

当 $\epsilon >= 0.5$ 时，上式不成立。随着集成中个体分类器数目 T 的增大，集成的错误率将指数级下降，最终趋向于零。

> 在现实中，个体学习器是解决同一个问题训练出来的，它们不可能相互独立，如何生成不同的个体学习器，是集成学习研究的核心。

根据个体学习器的生成方式，目前集成学习方法大致分为两大类：**个体学习期之间存在强依赖关系、必须串行生成的序列化方法—— Boosting；个体学习器之间不能存在强依赖关系、可同时生成的并行化方法—— Bagging。**

Boosting

> 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进行调整，使得先前基学习器出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。

AdaBoost

只适用二分类任务，比较容易理解的是基于“加性模型” additive model ，即基学习器的线性组合：

$$H(\boldsymbol{x})=\sum_{t=1}^{T}\alpha_t h_t(\boldsymbol{x})$$

最小化指数损失函数：

$$\ell_{\exp}(H|\mathcal{D})=\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H(\boldsymbol{x})}\right]$$

$f(x)$ 只有两个结果，1 或 -1 ：

$$\ell_{\exp}(H|\mathcal{D})=e^{-H(x)}P(f(x)=1)+e^{H(x)}P(f(x)=-1)$$

若 $H(x)$ 可以将损失函数最小化，那么对它求偏导：

$$\frac{\partial\ell_{\exp}(H|\mathcal{D})}{\partial H(\boldsymbol{x})}=-e^{-H(\boldsymbol{x})}P(f(\boldsymbol{x})=1|\boldsymbol{x})+e^{H(\boldsymbol{x})}P(f(\boldsymbol{x})=-1|\boldsymbol{x})$$

显然，令上式为0，可以解出：

$$H(\boldsymbol{x})=\frac{1}{2}\ln\frac{P(f(x)=1|\boldsymbol{x})}{P(f(x)=-1|\boldsymbol{x})}$$

因此：

$$\begin{aligned}\operatorname{sign}(H(\boldsymbol{x}))&=\operatorname{sign}\left(\frac{1}{2}\ln\frac{P(f(x)=1|\boldsymbol{x})}{P(f(x)=-1|\boldsymbol{x})}\right)\\&=\begin{cases}1, & P(f(x)=1|\boldsymbol{x})>P(f(x)=-1|\boldsymbol{x})\\-1, & P(f(x)=1|\boldsymbol{x})<P(f(x)=-1|\boldsymbol{x})\end{cases}\\&=\underset{y\in\{-1,1\}}{\arg\max}P(f(x)=y|\boldsymbol{x})\end{aligned}$$

我们发现，因为本身是二分类问题，特性非常优秀，指数损失函数最小化，则分类错误率也将最小，即达到了贝叶斯最优错误率。

因为我们的基分类器前面还有参数，当基分类器得到以后，该基分类器的权重 a_t 应该使得 $a_t h_t$ 最小化指数损失函数:

$$\begin{aligned}\ell_{\exp}\left(\alpha_t h_t|\mathcal{D}_t\right)&=\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[e^{-f(\boldsymbol{x})\alpha_t h_t(\boldsymbol{x})}\right]\\&=\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[e^{-\alpha_t}\mathbb{I}\left(f(\boldsymbol{x})=h_t(\boldsymbol{x})\right)+e^{\alpha_t}\mathbb{I}\left(f(\boldsymbol{x})\neq h_t(\boldsymbol{x})\right)\right]\\&=e^{-\alpha_i}P_{\boldsymbol{x}\sim\mathcal{D}_t}\left(f(\boldsymbol{x})=h_t(\boldsymbol{x})\right)+e^{\alpha_i}P_{\boldsymbol{x}\sim\mathcal{D}_t}\left(f(\boldsymbol{x})\neq h_t(\boldsymbol{x})\right)\\&=e^{-\alpha_t}\left(1-\epsilon_t\right)+e^{\alpha_t}\epsilon_t\end{aligned}$$

其中 $\epsilon_t=P_{x\sim\mathcal{D}_t}\left(h_t(\boldsymbol{x})\neq f(\boldsymbol{x})\right)$ ，考虑指数损失函数的导数：

$$\frac{\partial\ell_{\exp}\left(\alpha_t h_t|\mathcal{D}_t\right)}{\partial\alpha_t}=-e^{-\alpha_t}\left(1-\epsilon_t\right)+e^{\alpha_t}\epsilon_t$$

上式为0，可以得到**权重更新公式**：

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

AdaBoost 算法在下一轮基学习中纠正错误，那么：

$$\begin{aligned}\ell_{\exp}\left(H_{t-1}+h_t|\mathcal{D}\right) &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})(H_{t-1}(\boldsymbol{x})+h_t(\boldsymbol{x}))}\right] \\ &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}e^{-f(\boldsymbol{x})h_t(\boldsymbol{x})}\right]\end{aligned}$$

它可以进行泰勒展开，同时注意到 $f^2(x)=h_t^2(x)=1$：

$$\begin{aligned}\ell_{\exp}\left(H_{t-1}+h_t|\mathcal{D}\right) &\simeq \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\left(1-f(\boldsymbol{x})h_t(\boldsymbol{x})+\frac{f^2(\boldsymbol{x})h_t^2(\boldsymbol{x})}{2}\right)\right] \\ &= \mathbb{E}_{x\sim\mathcal{D}}\left[e^{-f(x)H_{t-1}(x)}\left(1-f(x)h_t(x)+\frac{1}{2}\right)\right]\end{aligned}$$

理想的基学习器：

$$\begin{aligned}h_t(\boldsymbol{x}) &= \arg\min_{\boldsymbol{h}}\ell_{\exp}\left(H_{t-1}+h|\mathcal{D}\right) \\ &= \arg\min_{h}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\left(1-f(\boldsymbol{x})h(\boldsymbol{x})+\frac{1}{2}\right)\right] \\ &= \arg\max_{h}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}f(\boldsymbol{x})h(\boldsymbol{x})\right] \\ &= \arg\max_{h}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[\frac{e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\right]}f(\boldsymbol{x})h(\boldsymbol{x})]\end{aligned}$$

因为 $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\right]$ 是一个常数，令 $\mathcal{D}_t$ 表示一个分布：

$$\mathcal{D}_t(\boldsymbol{x}) = \frac{\mathcal{D}(\boldsymbol{x})e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{w})}\right]}$$

这等价于令：

$$\begin{aligned}h_t(\boldsymbol{x}) &= \arg\max_{\boldsymbol{h}}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\frac{e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\right]}f(\boldsymbol{x})h(\boldsymbol{x})\right] \\ &= \arg\max_{\boldsymbol{h}}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}[f(\boldsymbol{x})h(\boldsymbol{x})]\end{aligned}$$

由于 $f(x),h(x)\in\{-1,+1\}$，有：

$$f(\boldsymbol{x})h(\boldsymbol{x}) = 1-2\mathbb{I}(f(\boldsymbol{x})\neq h(\boldsymbol{x}))$$

那么理想的基学习器：

$$h_t(\boldsymbol{x}) = \arg\min_{h}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}[\mathbb{I}(f(\boldsymbol{x})\neq h(\boldsymbol{x}))]$$

在分布 D_t 下最小化分类误差，**样本分布更新公式**：

$$
\begin{aligned}
\mathcal{D}_{t+1}(\boldsymbol{x}) &= \frac{\mathcal{D}(\boldsymbol{x}) e^{-f(\boldsymbol{x}) H_t(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}\left[e^{-f(\boldsymbol{x}) H_t(\boldsymbol{x})}\right]} \\
&= \frac{\mathcal{D}(\boldsymbol{x}) e^{-f(\boldsymbol{x}) H_{t-1}(\boldsymbol{x})} e^{-f(\boldsymbol{x}) \alpha_t h_t(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}\left[e^{-f(\boldsymbol{x}) H_t(\boldsymbol{x})}\right]} \\
&= \mathcal{D}_t(\boldsymbol{x}) \cdot e^{-f(\boldsymbol{x}) \alpha_t h_t(\boldsymbol{x})} \frac{\mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}\left[e^{-f(\boldsymbol{x}) H_{t-1}(\boldsymbol{x})}\right]}{\mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}\left[e^{-f(\boldsymbol{x}) H_t(\boldsymbol{x})}\right]}
\end{aligned}
$$

> 重赋权法：根据样本分布为每个样本重新赋予一个权重。
>
> 重采样法：根据样本分布对训练集重新采样，再用采样样本集对基学习器进行训练。
>
> Boosting 主要关注降低偏差，因此 Boosting 能基于泛化性能相当弱的学习器构建很强的集成。

AdaBoost 算法的误差分析

AdaBoost 能够在学习过程中不断减少训练误差，即在训练数据集上的分类误差，所以，有以下定理，AdaBoost 算法最终分类器的训练误差界（**定理1：AdaBoost 的训练误差界**）为：

$$
\frac{1}{N} \sum_{i=1}^{N} I\left(G\left(x_i\right) \neq y_i\right) \leqslant \frac{1}{N} \sum_{i} \exp\left(-y_i f\left(x_i\right)\right) = \prod_{m} Z_m
$$

这里的 $G(x)$ 就是我们的 $h_i(x)$ ，$f(x)$ 是正确结果，Z_m 是规范化因子。

首先，我们知道：

$$
w_{m+1,i} = \frac{w_{mi}}{Z_m} \exp\left(-\alpha_m y_i G_m\left(x_i\right)\right), \quad i = 1, 2, \cdots, N
$$

我们使 D_{m+1} 成为一个概率分布，通过：

$$
Z_m = \sum_{i=1}^{N} w_{mi} \exp\left(-\alpha_m y_i G_m\left(x_i\right)\right)
$$

所以可以推出：

$$
w_{mi} \exp\left(-\alpha_m y_i G_m\left(x_i\right)\right) = Z_m w_{m+1,i}
$$

上面式子的前半部分是显然的，当 $G(x_i) \neq y_i$, $y_i f(x_i) < 0$ 所以后面的结果一定大于 1 。然后的等号推导如下：

$$
\begin{aligned}
\frac{1}{N}\sum_i \exp(-y_i f(x_i)) &= \frac{1}{N}\sum_i \exp\left(-\sum_{m=1}^{M}\alpha_m y_i G_m(x_i)\right) \\
&= \sum_i w_{1i}\prod_{m=1}^{M}\exp(-\alpha_m y_i G_m(x_i)) \\
&= Z_1\sum_i w_{2i}\prod_{m=2}^{M}\exp(-\alpha_m y_i G_m(x_i)) \\
&= Z_1 Z_2\sum_i w_{3i}\prod_{m=3}^{M}\exp(-\alpha_m y_i G_m(x_i)) \\
&= \dots \\
&= Z_1 Z_2\cdots Z_{M-1}\sum_i w_M \exp(-\alpha_M y_i G_M(x_i)) \\
&= \prod_{m=1}^{N} z_m
\end{aligned}
$$

现在每一轮选取适当 G_m 使得 Z_m 最小，从而使训练误差下降最快，对二分类问题，有如下结果（**定理2：二分类问题 AdaBoost 的训练误差界**）：

$$
\prod_{m=1}^{M} Z_m = \prod_{m=1}^{M}\left[2\sqrt{e_m(1-e_m)}\right] = \prod_{m=1}^{M}\sqrt{(1-4\gamma_m^2)} \leqslant \exp\left(-2\sum_{m=1}^{M}\gamma_m^2\right)
$$

这里，$\gamma_m = \frac{1}{2} - e_m$ 。

前两个等号：

$$
\begin{aligned}
Z_m &= \sum_{i=1}^{N} w_{mi}\exp(-\alpha_m y_i G_m(x_i)) \\
&= \sum_{y_i = G_m(x_i)} w_{mi}\mathrm{e}^{-\alpha_m} + \sum_{y_i \neq G_m(x_i)} w_{mi}\mathrm{e}^{\alpha_n} \\
&= (1-e_m)\,\mathrm{e}^{-\alpha_m} + e_m\mathrm{e}^{\alpha_m} \\
&= 2\sqrt{e_m(1-e_m)} = \sqrt{1-4\gamma_m^2}
\end{aligned}
$$

然后，对于最后的不等号：

$$
\prod_{m=1}^{M}\sqrt{(1-4\gamma_m^2)} \leqslant \exp\left(-2\sum_{m=1}^{M}\gamma_m^2\right)
$$

可以通过 e^x 和 $\sqrt{1-x}$ 在点 $x=0$ 处泰勒展开得到。

推论：如果存在 $\gamma > 0$ ，对所有 m 有 $\gamma_m \geqslant \gamma$，则：

$$
\frac{1}{N}\sum_{i=1}^{N} I(G(x_i) \neq y_i) \leqslant \exp\left(-2M\gamma^2\right)
$$

AdaBoost 的训练误差以指数速率下降。

AdaBoost 算法的解释

考虑加法模型：

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

这里，$b(x;\gamma_m)$ 是基函数， γ_m 是基函数的参数，β_m 是基函数的系数，显然，这是一个加法模型。

在给定训练数据和损失函数 $L(y,f(x))$ 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：

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

给定训练数据集 $T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$ ，学习加法模型 $f(x)$ 的前向分布算法如下：

1. 初始化 $f_0(x)=0$
2. 循环开始 $m=1...M$
3. 极小化损失函数：$(\beta_m,\gamma_m)=\arg\min_{\beta,\gamma}\sum_{i=1}^{N}L\left(y_i,f_{m-1}\left(x_i\right)+\beta b\left(x_i;\gamma\right)\right)$
4. 得到参数 β_m 和 γ_m ，更新：$f_m(x)=f_{m-1}(x)+\beta_m b\left(x;\gamma_m\right)$
5. 最终，得到加法模型： $f(x)=f_M(x)=\sum_{m=1}^{M}\beta_m b\left(x;\gamma_m\right)$

现在，前向分布算法，将问题简化为逐次求解各个参数 β_m 和 γ_m 。

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。

AdaBoost 小结

训练数据中每个样本赋予一个权重，这个权重构成了向量 D ，之后分对的样本权重降低，分错的样本权重增高，构成新的 D ，同时 AdaBoost 为每个分类器都分配一个权重值 alpha：

$$\alpha=\frac{1}{2}\ln\left(\frac{1-\varepsilon}{\varepsilon}\right)$$

而分布 D：

$$D_i^{(t+1)}=\frac{D_i^{(t)}\mathrm{e}^{\pm\alpha}}{\mathrm{Sum}(D)}$$

进行下一轮迭代。

提升树

以决策树为基函数的提升方法被称为提升树，对分类问题决策树是二叉分类树，对回归问题决策树是二叉回归树。提升树可以表示为决策树的加法模型：

$$f_M(x)=\sum_{m=1}^{M}T\left(x;\Theta_m\right)$$

其中 $T\left(x;\Theta_n\right)$ 表示决策树；Θ_n 表示决策树的参数； M 是树的个数。

提升树算法采用**前向分步算法**。首先确定初始提升树 $f_0(x)=0$ ，第 m 步的模型是：

$$f_m(x) = f_{m-1}(x) + T(x;\Theta_m)$$

通过**经验风险极小化**确定下一棵决策树的参数：

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

> 这里的 T 指的就是下一棵决策树。

不同问题的提升树学习算法，主要区别在于使用的损失函数不同，平方误差损失函数的回归问题，指数损失函数的分类问题。下面叙述回归问题的提升树：

$$T(x;\Theta) = \sum_{j=1}^{J} c_j I\left(x \in R_j\right)$$

x 是输入， y 是输出，c 是输出常量，J 是回归树的复杂度即叶节点的个数，
$\Theta = \{(R_1, c_1), (R_2, c_2), \cdots, (R_J, c_J)\}$ 表示树的区域划分和各区域上的常数。

回归问题提升树使用以下前向分布算法：

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

在前向分布算法的第 m 步，给定当前模型 $f_{m-1}(x)$ ，需求解：

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

当使用平方误差损失函数时：

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

其损失变为：

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

这里， $r = y - f_{m-1}(x)$ ，是当前模型拟合数据的残差。

所以，对回归问题的提升树来说，只需要简单地拟合当前模型的残差，这样，算法就相当简单。

回归问题的提升树算法：

输入：训练数据集 $T = (x1, y1), (x2, y2), \ldots (xn, yn), xi, yi$

输出：提升树 $f_M(x)$

1. 初始化 $f_0(x) = 0$
2. 开始循环 m = 1,2,...M
3. 计算残差：$r_{mi} = y_i - f_{m-1}\left(x_i\right), \quad i = 1, 2, \cdots, N$
4. 拟合残差 r_{mi} 学习一个回归树，得到 $T\left(x;\Theta_m\right)$

5. 更新 $f_m(x) = f_{m-1}(x) + T(x; \Theta_m)$
6. 对第 3 步到第 5 步进行循环
7. 得到回归问题提升树 $f_M(x) = \sum_{m=1}^{M} T(x; \Theta_m)$

梯度提升

提升树利用加法模型和前向分布算法实现学习的优化过程，当损失函数是平方损失和指数损失的时候，每一步优化是很简单的，但一般损失函数而言，往往每一步优化并不容易，针对这一问题，出现了梯度提升。

这是利用最速下降法的近似方法，其关键是利用损失函数的负梯度在当前模型的值，主要不同就是残差的计算方式：

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

作为回归问题提升树算法中的残差的近似值。

输入：训练数据集 $T = (x1, y1), (x2, y2), \ldots (xn, yn), xi, yi$； 损失函数 $L(y, f(x))$

输出：回归树 $\hat{f}(x)$

1. 初始化：$f_0(x) = \arg\min_c \sum_{i=1}^{N} L(y_i, c)$
2. 开始循环 m 从 1 到 M
3. 对于 i 从 1 到 N，计算：$r_{ml} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)}$
4. 对 r_{mi} 拟合一个回归树，得到第 m 棵树的叶节点区域 R_{mj}
5. 对 j 从 1 到 J，计算：$c_{mj} = \arg\min_c \sum_{x_i \in R_{mj}} L(y_i, f_{m-1}(x_i) + c)$
6. 更新 $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J} c_{mj} I(x \in R_{mj})$
7. 得到回归树 $\hat{f}(x) = f_M(x) = \sum_{m=1}^{M} \sum_{j=1}^{J} c_{mj} I(x \in R_{mj})$

XGBoost

原始的 GBDT 算法基于经验损失函数的负梯度来构造新的决策树，只是在决策树构建完成后再进行剪枝，而 XGBoost 在决策树构建阶段就加入了正则化：

$$L_t = \sum_{i=1}^{n} l(y_i, F_{t-1}(x_i) + f_t(x_i)) + \sum_{k=1}^{K} \Omega(f_k)$$

对于上面的式子，我们可以发现，除去正则项以外，就是我们传统的决策树。对于决定下一棵树：

$$\begin{aligned}
\text{obj}^{(t)} &= \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t)}\right) + \sum_{i=1}^{t} \Omega(f_i) \\
&= \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + \text{ constant}
\end{aligned}$$

现在我们使用泰勒展开， x 取值 $\hat{y}_i^{(t-1)} + f_t(x_i)$ ，来逼近：

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

其中：

$$g_i = \partial_{\hat{y}_i(t-1)} l\left(y_i, \hat{y}_i^{(t-1)}\right)$$
$$h_i = \partial^2_{\hat{y}_i^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right)$$

删除常数项，那么 t 目标函数就变成了：

$$\sum_{i=1}^{n}\left[g_i f_t\left(x_i\right)+\frac{1}{2} h_i f_t^2\left(x_i\right)\right]+\Omega\left(f_t\right)$$

我们需要定义树的复杂度 $\Omega(f)$ ，首先我们定义一棵树：

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

这里 w 是树叶上的分数向量，q 是将每个数据点分配给叶子的函数，T 是树叶的数量。正则化定义：

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

注意，当正则项系数为 γ 为 0 时，整体目标就退化回了 GBDT 。

我们可以用第 t 棵树来编写目标值如：

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

其中 $I_j=\{i \mid q\left(x_i\right)=j\}$ 是分配给第 j 个叶子的数据点的索引的集合。 请注意，在第二行中，我们更改了总和的索引，因为同一叶上的所有数据点都得到了相同的分数。 我们可以通过定义 $G_j=\sum_{i \in I_j} g_i$ 和 $H_j=\sum_{i \in I_j} h_i$ 来进一步压缩表达式：

$$Obj^{(t)}=\sum_{j=1}^{T}\left[G_j w_j+\frac{1}{2}\left(H_j+\lambda\right) w_j^2\right]+\gamma T$$

我们可以得到最好的客观规约：

$$w_j^*=-\frac{G_j}{H_j+\lambda}$$

将预测值带入损失函数可以得到损失函数的最小值，同时也在度量一个树有多好：

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

既然我们有了一个方法来衡量一棵树有多好，理想情况下我们会列举所有可能的树并挑选出最好的树。 在实践中，这种方法是比较棘手的，所以我们会尽量一次优化树的一个层次。 具体来说，我们试图将一片叶子分成两片，并得到分数：

$$\text{Gain} = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda}+\frac{G_R^2}{H_R+\lambda}-\frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right]-\gamma$$

这个公式可以分解为 1) 新左叶上的得分 2) 新右叶上的得分 3) 原始叶子上的得分 4) additional leaf（附加叶子）上的正则化。 我们可以在这里看到一个重要的事实：如果增益小于 γγ，我们最好不要添加那个分支。这正是基于树模型的 **pruning（剪枝）** 技术！通过使用监督学习的原则，我们自然会想出这些技术工作的原因 :)

另外，在分割的时候，这个系统还能感知稀疏值，我们给每个树的结点都加了一个默认方向，当一个值是缺失值时，我们就把他分类到默认方向，每个分支有两个选择，具体应该选哪个？这里提出一个算法，枚举向左和向右的情况，哪个 gain 大选哪个，这些都在这里完成。

总结一下，XGBoost 就是最大化这个差来进行决策树的构建，XGBoost 和 GDBT 的差别和联系：

- GDBT 是机器学习算法， XGBoost 是该算法的工程实现。
- XGBoost 加入了正则化，支持多种类型的基分类器，支持对数据采样（和 RF 类似），能对缺省值处理。

> ps: 论文第二章里提到了shrinkage 和 column subsampling，就是相当于学习速率和对于列的采样骚操作。**调低 eta 能减少个体的影响，给后续的模型更多学习空间**。对于列的重采样，根据一些使用者反馈，列的 subsampling 比行的 subsampling 效果好，列的 subsampling 也加速了并行化的特征筛选。

XGBoost 的调参

- 过拟合：

> 直接控制模型的复杂度：
>
> - 这包括 `max_depth`, `min_child_weight` 和 `gamma`
>
> 增加随机性，使训练对噪声强健：
>
> - 这包括 `subsample`, `colsample_bytree`
> - 你也可以减小步长 `eta`, 但是当你这么做的时候需要记得增加 `num_round` 。

- 不平衡的数据集

> 如果你只关心预测的排名顺序：
>
> - 通过 `scale_pos_weight` 来平衡 positive 和 negative 权重。
> - 使用 AUC 进行评估
>
> 如果你关心预测正确的概率：
>
> - 在这种情况下，您无法重新平衡数据集
> - 在这种情况下，将参数 `max_delta_step` 设置为有限数字（比如说1）将有助于收敛

算法实现

AdaBoost 伪码

```
"""
训练集 D = {(x1, y1), (x2, y2)..., (xm, ym)}
基学习算法 L
训练轮数 T
"""
D[1] = 1/m # 初始化样本权值分布
for t in range(T):
  h[t] = L(D, D[t]) # 基于分布 Dt 从数据集 D 中训练处分类器 ht
  e[t] = P(ht(x), f(x)) # 分类器 ht 的误差， ht(x) 是预测结果， f(x) 是真实结果
  if e[t] > 0.5:
    break
  a[t] = 0.5*np.log((1-e[t])/e[t])
  D[t+1] = D[t] / Z[t] # Zt 是规范化因子
  if (h[t](x) == f(x)) D[t+1] *= exp(-a[t]) # 更新 D[t+1] 的权重
  else D[t+1] *= exp(a[t])
```

最终返回 $H(x)=\operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right)$ 。

> sign 表达符号函数。

算法例子

AdaBoost 例子

假设存在3个分类器，对每一个分类器：

- 初始化权值 Di。
- 取阈值来分类，得到基分类器 hi。
- 计算误差率 ei。
- 得到分类器系数 ai。
- 更新权值 Di+1。

最后我们将三个分类器按照各自的系数 a 来进行预测，得到整体 H 。

如果没看懂我们再来一次：

输入数据集 $T=\{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$ 。

输出最终分类器 $G(x)$ 。

> Ps：刚才我们用 H 来表示分类器。

1. 初始化训练数据的权值分布：$D_1=(w_{11}, \cdots, w_{1i}, \cdots, w_{1N}), \quad w_1=\frac{1}{N}, \quad i=1,2, \cdots, N$
2. 循环开始，对于 $m=1,2, \ldots M$
3. 使用具有权值分布 D_m 的训练数据学习，得到基本分类器：$G_m(x): \mathcal{X} \rightarrow\{-1,+1\}$
4. 计算 $G_m(x)$ 在训练数据集上的分类误差率：
   $e_m=P\left(G_m\left(x_i\right) \neq y_i\right)=\sum_{i=1}^{N} w_{mi} I\left(G_m\left(x_i\right) \neq y_i\right)$
5. 计算 $G_m(x)$ 的系数：$\alpha_m=\frac{1}{2} \log \frac{1-e_m}{e_m}$
6. 更新训练集的权值分布：$D_{m+1}=\left(w_{m+1,1}, \cdots, w_{m+1, l}, \cdots, w_{m+1, N}\right)$

$w_{m+1,i}=\frac{w_{mi}}{Z_m}\exp(-\alpha_m y_i G_m\left(x_i\right)), \quad i=1,2,\cdots,N$

7. 这里的 Z_m 是规范化因子：$Z_m=\sum_{i=1}^{N} w_m \exp(-\alpha_m y_i G_m\left(x_i\right))$ ，它使 D_{m+1} 成为一个概率分布
8. 构建基本分类器的线性组合：$f(x)=\sum_{m=1}^{M} \alpha_m G_m(x)$
9. 得到最终分类器：$G(x)=\operatorname{sign}(f(x))=\operatorname{sign}\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$

经典题目

- XGBoost 与 GBDT 的联系与区别有哪些?

> GBDT 是机器学习算法；XGBoost 是工程实现。
>
> 传统 GBDT 采用 CART 作为基分类器， XGBoost 支持多种类型的基分类器，比如线性分类器。
>
> XGBoost 增加了正则项，防止过拟合。
>
> XGBoost 支持对数据进行采样，对缺失值有处理。

从方差和偏差的角度解释 Boosting 和 Bagging 的原理?

算法总结

- 提升算法是将弱学习算法提升为强学习算法的统计学习算法，通过反复修改训练数据的权值分布，构建一系列基本分类器，并将这些基本分类器线性组合，构成一个强分类器。
- AdaBoost 将分类误差小的基本分类器以大的权值，给误差大的基本分类器以小的权值。
- 提升树是以分类树或回归树为基本分类器的提升方法。