

提升方法 (Boosting)

1. 提升方法与 AdaBoost 算法

提升方法: 多个弱分类器组合成强分类器 (三个臭皮匠, 顶个诸葛亮)。

Adaboost: 提高那些被前一轮弱分类器错误分类的样本的权值, 降低被正确分类的样本的权值; 通过加权多数表决的方法得到分类结果。

1.1. AdaBoost 算法

输入: 训练数据集 $\{(x_1, y_1), \dots, (x_N, y_N)\}$, $y_i \in \{-1, 1\}$

输出: 最终分类器 $G(x)$

(1) 初始化权值分布

$$D_1 = (w_{11}, \dots, w_{1N})^\top, \quad w_{1i} = \frac{1}{N} \quad (1.1)$$

(2) 对 $m = 1, \dots, M$,

(a) 使用有权值分布 D_m 的训练数据集学习, 得到分类器 $G_m(x)$ (取值 $\{-1, 1\}$)

(b) 计算 $G_m(x)$ 在训练数据集上的分类误差率:

$$e_m = \sum_i w_{mi} I\{G_m(x_i) \neq y_i\} \quad (1.2)$$

(c) 计算 $G_m(x)$ 的系数:

$$\alpha_m = \frac{1}{2} \log \left\{ \frac{1 - e_m}{e_m} \right\} \quad (1.3)$$

(d) 更新训练数据集的权值分布

$$w_{m+1,i} = \frac{w_{mi}}{Z_m} \exp(-\alpha_m y_i G_m(x_i)) \quad (1.4)$$

其中 $Z_m = \sum_i w_{mi} \exp(-\alpha_m y_i G_m(x_i))$

(3) 令 $f(x) = \sum_m \alpha_m G_m(x)$, 最终得到分类器 $G(x) = \text{sign}(f(x))$.

注:

(1) α_m 随着 e_m 的递减而递增 (对于 $e_m < 1/2$).

(2) $w_{m+1,i}$ 对于错分类样本权重会增加

请修改例 8.1 的数据, 作为一个作业

1.2. AdaBoost 的训练误差分析

Theorem 1. (AdaBoost 的训练误差边界) AdaBoost 算法最终分类器的训练误差界为:

$$\frac{1}{N} \sum_i I(G(x_i) \neq y_i) \leq \frac{1}{N} \sum_i \exp(-y_i f(x_i)) = \prod_m Z_m,$$

其中 $w_{mi} = \exp(-y_i f_{m-1}(x_i)) / \sum_i \exp(-y_i f_{m-1}(x_i))$, $Z_m = \sum_i w_{mi} \exp(-\alpha_m y_i G_m(x_i))$.

解读: 可以在每一轮选取适当的 G_m 使得 Z_m 最小, 从而使得训练误差下降最快。

证明: (1) 第一个不等式证明比较简单。注意到 $\exp(-y_i f(x_i)) > 1$ 对于错分类样本 ($f(x_i)y_i < 0$)。因此左式成立。

(2) 后半部分推导如下: 注意到 Adaboost 中用到的权重更新公式:

$$w_{mi} \exp\{-\alpha_m y_i G_m(x_i)\} = Z_m w_{m+1,i} \quad (1.5)$$

则可得以下结论:

$$\begin{aligned} \frac{1}{N} \sum_i \exp(-y_i f(x_i)) &= \frac{1}{N} \sum_i \exp(-y_i \sum_m \alpha_m G_m(x_i)) \\ &= \sum_i w_{1i} \prod_{m=1}^M \exp(-y_i \alpha_m G_m(x_i)) = Z_1 \sum_i w_{2i} \prod_{m=2}^M \exp(-y_i \alpha_m G_m(x_i)) = \prod_{m=1}^M Z_m. \end{aligned}$$

Theorem 2. (二分类问题的 *AdaBoost* 训练误差界)

$$\prod_{m=1}^M Z_m = \prod_{m=1}^M \{2\sqrt{e_m(1-e_m)}\} = \prod_{m=1}^M \sqrt{1-4\gamma_m^2} \leq \exp(-2 \sum_{m=1}^M \gamma_m^2).$$

这里 $\gamma_m = 1/2 - e_m$.

解读：如果存在 $\gamma > 0$ ，对所有的 $\gamma_m \geq \gamma$ ，则有

$$\frac{1}{N} \sum_i I\{G(x_i) \neq y_i\} \leq \exp(-2M\gamma^2).$$

证明：由 Z_m 的定义知：

$$\begin{aligned} Z_m &= \sum_m w_{mi} \exp(-\alpha_m y_i G_m(x_i)) = \sum_{y_i=G_m(x_i)} w_{mi} \exp(-\alpha_m) + \sum_{y_i \neq G_m(x_i)} w_{mi} \exp(\alpha_m) \\ &= (1 - e_m) \exp(-\alpha_m) + e_m \exp(\alpha_m) \end{aligned}$$

注意到

$$\alpha_m = \frac{1}{2} \log \frac{1 - e_m}{e_m}$$

则上式等于

$$2\sqrt{e_m(1-e_m)} = \sqrt{1-4\gamma_m^2}.$$

比较 $\sqrt{1-2x}$ 及 $\exp(-x)$ 的函数图像，则后半部分可得。

1.3. *AdaBoost* 算法的解读

(1) 前向分步算法

可加模型：

$$f(x) = \sum_{m=1}^M \beta_m b(x; \gamma_m) \tag{1.6}$$

给定损失函数 $L(y, f(x))$, 则需优化

$$\min_{\beta_m, \gamma_m} \sum_i L\left(y_i, \sum_m \beta_m b(x_i; \gamma_m)\right) \quad (1.7)$$

算法 1 (前向分步算法)

输入: 训练数据集 $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$; 损失函数 $L(y, f(x))$; 基函数集 $\{b(x; \gamma)\}$;

输出: 加法模型 $f(x)$;

(1) 初始化 $f_0(x) = 0$;

(2) 对 $m = 1, 2, \dots, M$;

(a) 极小化损失函数

$$(\beta_m, \gamma_m) = \arg \min_{\beta, \gamma} \sum_{i=1}^N L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)) \quad (1.8)$$

得到参数 β_m, γ_m ;

(b) 更新

$$f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m) \quad (1.9)$$

(3) 得到加法模型

$$f(x) = f_M(x) = \sum_{m=1}^M \beta_m b(x; \gamma_m) \quad (1.10)$$

这样, 前向分步算法将同时求解从 $m=1$ 到 M 所有的参数 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。

(2) 前向算法与 **AdaBoost**

Theorem 3. *AdaBoost* 算法是前向分步算法的特例。此时, 模型是基本模型组成的

加法模型，损失函数是指数函数。其中，指数损失函数形式为：

$$L(y, f(x)) = \exp \{ -yf(x) \} \quad (1.11)$$

Proof:

Guideline: 需要证明 α_m 的计算公式，及权重更新的公式

假设经过 $m-1$ 轮迭代已经得到 $f_{m-1}(x)$:

$$f_{m-1}(x) = \sum_{i=1}^{m-1} \alpha_i G_i(x) \quad (1.12)$$

在第 m 轮需要迭代得到 α_m , $G_m(x)$ 和 $f_m(x)$.

$$\begin{aligned} (\alpha_m, G_m(x)) &= \arg \min_{\alpha, G} \sum_i \exp \{ -y_i(f_{m-1}(x_i) + \alpha G(x)) \} \\ &= \arg \min_{\alpha, G} \sum_i \bar{w}_{mi} \exp \{ -y_i \alpha G(x_i) \} \end{aligned}$$

求解上式。首先，求 $G_m^*(x)$. 对任意的 $\alpha > 0$, 最小的 $G(x)$ 由下式得到

$$G_m^* = \arg \min_G \sum_i \bar{w}_{mi} I(y_i \neq G(x_i)) \quad (1.13)$$

注：这是由于

$$\sum_i \bar{w}_{mi} \exp(-y_i \alpha G(x_i)) = \sum_{y_i \neq G(x_i)} \bar{w}_{mi} \exp(\alpha) + \sum_{y_i = G(x_i)} \bar{w}_{mi} \exp(-\alpha) \quad (1.14)$$

因此 $G_m(x)$ 为使得第 m 轮加权训练数据分类误差最小的基本分类器。

然后，求解 α , 将式子对 α 求导并使之为零，可得：

$$\alpha_m = \frac{1}{2} \log \frac{1 - e_m}{e_m} \quad (1.15)$$

其中

$$e_m = \frac{\sum_i \bar{w}_{mi} I(y_i \neq G_m(x_i))}{\sum_i \bar{w}_{mi}} = \sum_i w_{mi} I(y_i \neq G_m(x_i)) \quad (1.16)$$

同时

$$\bar{w}_{m+1,i} = \bar{w}_{mi} \exp\{-y_i \alpha_m G_m(x_i)\} \quad (1.17)$$

这与 AdaBoost 的样本权重更新等价（只差一个规范化因子）。

2. 提升树

2.1. 提升树模型

以决策树为基的提升方法称为提升树（boosting tree）。提升树可以表示为决策树的加法模型：

$$f_M(x) = \sum_m T(x; \Theta_m) \quad (2.1)$$

(1) 提升树算法

提升树第 m 步的模型为：

$$f_m(x) = f_{m-1}(x) + T(x; \Theta_m) \quad (2.2)$$

通过经验风险极小化求解下一棵决策树参数 Θ_m ：

$$\hat{\Theta}_m = \arg \min_{\Theta_m} \sum_i L(y_i, f_{m-1}(x_i) + T(x_i; \Theta_m)) \quad (2.3)$$

对二类分类问题：只需将 Adaboost 中的基本分类器设为二类分类树。

对于回归问题，采用平方误差，则有：

$$L(y, f_{m-1}(x) + T(x; \Theta_m)) = \{y - f_{m-1}(x) - T(x; \Theta_m)\}^2 \quad (2.4)$$

相当于对残差进行拟合。

请适当修改例 8.2 变成一个作业

2.2. 梯度提升

对于一般损失函数而言，每一步优化并不容易。Freidman 提出了梯度提升的概念。关键是利用损失函数的负梯度在当前模型的值：

$$-\left[\frac{\partial L(y, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)} \quad (2.5)$$

算法 2 (梯度提升算法)

输入: 训练数据集 $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$, $x_i \in \mathcal{X} \subseteq \mathbf{R}^n$, $y_i \in \mathcal{Y} \subseteq \mathbf{R}$;
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) 对 $m = 1, 2, \dots, M$

(a) 对 $i = 1, 2, \dots, N$, 计算

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

(b) 对 r_{mi} 拟合一个回归树, 得到第 m 棵树的叶节点区域 $R_{mj}, j = 1, 2, \dots, J$;

(c) 对 $j = 1, 2, \dots, J$, 计算

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

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

(3) 得到回归树

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

算法第 1 步初始化，估计使损失函数极小化的常数值，它是只有一个根结点的数。第 2(a) 步计算损失函数的负梯度在当前模型的值，将它作为残差的估计。对于平方损失函数，它就是通常所说的残差；对于一般损失函数，它就是残差的近似值。第 2(b) 步估计回归树叶结点区域，以拟合残差的近似值。第 2(c) 步利用线性搜索估计叶结点区域的值，使损失函数极小化。第 2(d) 步更新回归树。第 3 步得到输出的最终模型 $\hat{f}(x)$ 。