第六章　集成学习

张皓

https://haomood.github.io/homepage/

zhangh0214@gmail.com

摘要

集成学习对多个学习器的预测进行结合以提升集成性能. 本章对集成学习进行简要介绍, 包括设计集成学习算法的关键—多样性, 以及集成学习的两大代表性方法: *Bagging* 和 *Boosting*. 在 *Boosting* 系列方法中, 本章介绍使用最广泛的 *AdaBoost* 和 *GradientBoosting*.

本系列文章有以下特点: *(a)*. 为了减轻读者的负担并能使尽可能多的读者从中收益, 本文试图尽可能少地使用数学知识, 只要求读者有基本的微积分、线性代数和概率论基础, 并在第一节对关键的数学知识进行回顾和介绍. *(b)*. 本文不省略任何推导步骤, 适时补充背景知识, 力图使本节内容是自足的, 使机器学习的初学者也能理解本文内容. *(c)*. 机器学习近年来发展极其迅速, 已成为一个非常广袤的领域. 本文无法涵盖机器学习领域的方方面面, 仅就一些关键的机器学习流派的方法进行介绍. *(d)*. 为了帮助读者巩固本文内容, 或引导读者扩展相关知识, 文中穿插了许多问题, 并在最后一节进行问题的 "快问快答".

1　集成学习基础

1.1　个体与集成

定义 1 (集成学习 (ensemble learning))**.** 构建并结合多个个体学习器 (individual learner) 来完成学习任务. 如果集成中只包含同种类型的个体学习器, 这样的集成是同质 (homogeneous) 的, 此时个体学习器也称为基学习器 (base learner). 如果集成包含不同类型的个体学习器, 这样的集成是异质 (heterogeneous) 的, 此时个体学习器也称为组件学习器 (component learner).

集成学习的好处? 有三个方面的好处 [6]. (1). 从统计的方面看, 由于学习任务的假设空间往往很大, 可能有多个假设在训练集上达到同等性能, 此时若使用单学习器可能因误选而导致泛化性能不佳, 结合多个学习器则会减小这一风险. (2). 从计算的方面看, 学习算法往往会陷入局部极小, 有的局部极小点所对应的泛化性能可能很糟糕, 而通过多次运行之后进行结合, 可降低陷入糟糕局部极小点的风险. (3). 从表示的方面看, 某些学习任务的真实假设可能不在当前学习算法所考虑的假设空间中, 此时若使用单学习器肯定无效, 而通过结合多个学习器, 由于相应的假设空间有所扩大, 有可能学得更好的近似.

为什么集成学习通常研究弱学习器? 弱学习器 (weak learner) 是指泛化性能略优于随机猜测的学习器. 通过将多个弱学习器进行集成, 常可获得比单一学习器显著优越的泛化性能, 因此集成学习的很多理论研究都是针对弱学习器进行的. 但在实践中为了使用较少的学习器, 或重用关于常见学习器的一些经验等, 人们往往会使用比较强的学习器.

1.2　基学习器结合策略

假设集成包含 T 个基学习器 $h_1, h_2, \dots, h_T$, 集成结果是 H. 对 h_t 进行结合的常见策略如表 1 .

为什么要约束 $\alpha_t \geq 0$? Breiman [3] 在研究 Stacking 回归时发现, 必须使用非负权重才能确保集成性能优于单一最佳个体学习器. 因此在集成学习中一般对学习器的权重施以非负约束.

Table 1: 基学习器结合策略. 其中 $h_t(\boldsymbol{x})_c$ 是基学习器 $h_t(\boldsymbol{x})$ 在类别 c 上的输出.

策略	方式
简单平均	$H(\boldsymbol{x}) := \frac{1}{T}\sum_{t=1}^{T} h_t(\boldsymbol{x})$
加权平均	$H(\boldsymbol{x}) := \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})$, 其中 $\forall t.\ \alpha_t \geq 0$ 以及 $\sum_{t=1}^{T} \alpha_t = 1$
绝对多数投票	$H(\boldsymbol{x}) := \begin{cases} c & \text{若 } \sum_{t=1}^{T} h_t(\boldsymbol{x})_c > \frac{1}{2}\sum_{c=1}^{C}\sum_{t=1}^{T} h_t(\boldsymbol{x})_c \\ \text{拒绝预测} & \text{否则} \end{cases}$, 即若某标记得票过半数, 则预测为该标记
相对多数投票	$H(\boldsymbol{x}) := \arg\max_c \sum_{t=1}^{T} h_t(\boldsymbol{x})_c$, 即预测得票最多的标记
加权投票	$H(\boldsymbol{x}) := \arg\max_c \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})_c$, 其中 $\forall t.\ \alpha_t \geq 0$ 以及 $\sum_{t=1}^{T} \alpha_t = 1$
Stacking [2, 26]	通过另一个学习器, 称为次级学习器或元学习器 (meta-learner), 对个体学习器 (称为初级学习器) 的结果进行结合

简单平均和加权平均的对比? 简单平均是加权平均在 $\alpha_t = \frac{1}{T}$ 的特例. 一般的, 加权平均的权重一般是从训练数据中学习而得, 例如估计出个体学习器的误差, 之后令权重大小与误差大小成反比. 现实任务中训练样本通常不充分或存在噪声, 这将使得学出的权重不完全可靠. 尤其是对规模比较大的集成来说, 要学习的权重比较多, 较容易导致过拟合. 因此, 实验和应用都显示出, 加权平均未必一定优于简单平均 [13, 15, 28]. 一般而言, 在个体学习器性能相差较大时宜使用加权平均, 而在个体学习器性能相近时宜使用简单平均.

绝对多数投票和相对多数投票的对比? 标准的绝对多数投票提供了 "拒绝预测" 选项, 这在可靠性要求较高的学习任务中是一个很好的机制. 但若学习任务要求必须提供预测结果, 则绝对多数投票将退化为相对多数投票.

硬投票和软投票各自的含义及适用场合? 硬投票 (hard voting) 是指用类标记 $h_t(\boldsymbol{x})_c \in \{0, 1\}$ 进行投票, 其中若 h_t 将样本 $\boldsymbol{x}$ 预测为标记 c 则取值为 1, 否则为 0. 软投票 (soft voting) 是指用类概率 $h_t(\boldsymbol{x})_c \in [0, 1]$ 进行投票, 其中 $h_t(\boldsymbol{x})_c$ 相当于对后验概率 $\Pr(y_i = c \mid \boldsymbol{x})$ 的估计. 硬投票和软投票不能混用. 对于异质集成, 由于基学习器类型不同, 类概率值不能直接比较, 此时适用于硬投票. 对于同质集成, 虽然估计的类概率值一般都不太准确, 但是基于类概率进行结合却往往比直接基于类标记进行结合性能更好. 对一些能在预测类别标记同时产生分类置信度的学习器 (如支持向量机的分类间隔), 其分类置信度可通过 Platt 缩放 [20] 、等分回归 (isotonic regresssion) [29] 等校准 (calibration) 后作为类概率使用.

Stacking 的具体过程? 包括如下三步.

1. 利用训练集 $D := \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$ 分别训练初级学习器 $h_1, h_2, \ldots, h_T$.
2. 构造次级训练集 $\tilde{D} := \{(\tilde{\boldsymbol{x}}_i, y_i)\}_{i=1}^{m}$, 其中

$$\tilde{\boldsymbol{x}}_i := \begin{bmatrix} h(\boldsymbol{x}_i)_1 \\ h(\boldsymbol{x}_i)_2 \\ \vdots \\ h(\boldsymbol{x}_i)_T \end{bmatrix} \in \mathbb{R}^T . \tag{1}$$

3. 利用次级训练集 $\tilde{D}$ 训练次级学习器 H.

如果降低 Stacking 的过拟合风险? 如果直接使用 (初级) 训练集 D 生成次级训练集 $\tilde{D}$, 则过拟合风险比较大. 因此, 一般使用交叉验证或者留一法, 在每一折, 用训练初级学习器未使用的样本来产生次级学习器的训练样本. 在这个交叉验证过程结束后, 产生次级训练集 $\tilde{D}$, 大小为 m.

Stacking 的次级学习器如何选择? 次级学习器的输入属性表示和次级学习算法对 Stacking 集成的泛化性能有很大影响. 有研究表明, 将初级学习器的输出类概率作为次级学习器的输入属性, 用多响应回归 (multi-response linear regression, MLR) 作为次级学习算法效果较好 [25] . MLR 对分类的每个类别进行线性回归, 属于对应标记的输出被置为 1, 其他类置为 0. 测试示例被分给输出值最大的类.

1.3 多样性

引理 1. *假设基分类器的错误率都为 e 且相互独立, 对二分类问题, 简单平均集成的错误率随个体分类器数目*

	测试例1	测试例2	测试例3
h_1	√	√	×
h_2	×	√	√
h_3	√	×	√
集成	√	√	√

(a) 集成提升性能

	测试例1	测试例2	测试例3
h_1	√	√	×
h_2	√	√	×
h_3	√	√	×
集成	√	√	×

(b) 集成不起作用

	测试例1	测试例2	测试例3
h_1	√	×	×
h_2	×	√	×
h_3	×	×	√
集成	×	×	×

(c) 集成起负作用

Figure 1: 集成个体应 “好而不同”. 图中 h_i 表示第 i 个分类器, √ 表示分类正确, × 表示分类错误. 本图源于 [32].

T 指数下降

$$\mathbb{E}[\mathbb{I}(H(\boldsymbol{x}) \neq y)] \leq \exp\left(-\frac{1}{2}T(1-2e^2)\right). \tag{2}$$

Proof. 若有超过半数的基分类器分类正确, 则集成分类就正确. 利用 Hoeffding 不等式,

$$\Pr(H(\boldsymbol{x}) \neq y) = \sum_{k=0}^{\lfloor \frac{T}{2} \rfloor} \binom{T}{k} (1-e)^k e^{T-k} \leq \exp(-\frac{1}{2}T(1-2e^2)). \tag{3}$$

□

引理 1 假设基学习器误差相互独立, 但在现实任务中, 基学习器是为解决同一个问题训练出来的, 它们显然不独立. 通过在学习过程引入随机性, 可以获得依赖程度没有那么高的学习器.

定理 2 (误差-分歧分解 (error-ambiguity decomposition) [16]). 假设使用加权平均法完成回归任务, 则

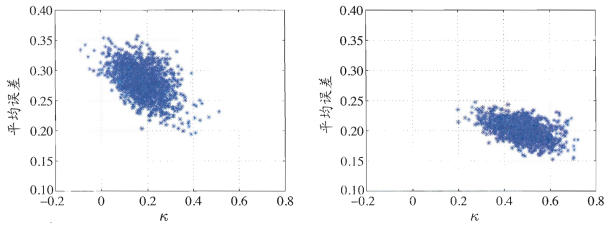
$$(H(\boldsymbol{x})-y)^2 = \sum_{t=1}^{T} \alpha_t (h_t(\boldsymbol{x})-y)^2 - \sum_{t=1}^{T} \alpha_t (h_t(\boldsymbol{x})-H(\boldsymbol{x}))^2 . \tag{4}$$

可以看出, 个体学习器准确性越高、多样性越大, 则集成越好.

Proof. 以下简记 $H(\boldsymbol{x})$ 为 H, $h_t(\boldsymbol{x})$ 为 h_t,

$$\begin{aligned}
(H-y)^2 &= y^2 - 2yH + H^2 \\
&= \sum_{t=1}^{T} \alpha_t h_t^2 - 2Hy + y^2 - \sum_{t=1}^{T} \alpha_t h_t^2 + 2H^2 - H^2 \\
&= \sum_{t=1}^{T} \left(\alpha_t h_t^2 - 2h_t y + y^2\right) - \sum_{t=1}^{T} \left(\alpha_t h_t^2 - 2h_t H + H^2\right) \\
&= \sum_{t=1}^{T} \alpha_t (h_t - y)^2 - \sum_{t=1}^{T} \alpha_t (h_t - H)^2 .
\end{aligned}$$

□



(a) AdaBoost 集成　　(b) Bagging 集成

Figure 2: 在 UCI 数据集 tic-tac-toc 上的 *kappa*-误差图. 每个集成包含 50 棵 C4.5 决策树. 本图源于 [32] .

误差-分歧分解的启示? 为了得到使集成结果比单一学习器更好, 个体学习器要做到 “好而不同”, 即个体学习器至少不差于弱学习器, 并且要有多样性 (diversity), 学习器之间具有差异, 如图 1 所示. 事实上, 个体学习器的准确性和多样性本身就存在冲突. 一般的, 准确性很高之后, 要增加多样性就需要牺牲准确性.

多样性的度量方法? 基本思路是考虑个体分类器两两之间的相似性, 对二分类数据集 D, 根据分类器 h_t 和 h_k 的预测结果的不同, 定义

$$a := \sum_{i=1}^{m} \mathbb{I}(h_t(\boldsymbol{x}_i) = +1 \wedge h_k(\boldsymbol{x}_i) = +1) ; \tag{5}$$

$$b := \sum_{i=1}^{m} \mathbb{I}(h_t(\boldsymbol{x}_i) = +1 \wedge h_k(\boldsymbol{x}_i) = -1) ; \tag{6}$$

$$c := \sum_{i=1}^{m} \mathbb{I}(h_t(\boldsymbol{x}_i) = -1 \wedge h_k(\boldsymbol{x}_i) = +1) ; \tag{7}$$

$$d := \sum_{i=1}^{m} \mathbb{I}(h_t(\boldsymbol{x}_i) = -1 \wedge h_k(\boldsymbol{x}_i) = -1) , \tag{8}$$

它们满足 $a+b+c+d = m$. 常见的多样性度量如表 2 所示. 此外, 可以将个体学习器成对的平均误差和多样性度量绘制成二维散点图, 如图 2 所示. 散点位置越低, 这对分类器的准确性越高; 散点位置越靠左, 这对分类器的多样性越大. 事实上, 现有的多样性度量都存在显

Table 2: 多样性度量方法. 其中, p_1 是两个分类器取得一致的概率, 估算为 $p_1 := \frac{a+d}{m}$; p_2 是两个分类器偶然达成一致的概率, 估算为 $p_2 := \frac{(a+b)(a+c)+(c+d)(b+d)}{m^2}$.

度量	定义	与多样性的关系
不合度量 (disagreement measure)	$d := \frac{b+c}{m} \in [0,1]$	值越大多样性越大
相关系数 (correlation coefficient)	$\rho := \frac{ad-bc}{\sqrt{(a+b)(a+c)(c+d)(b+d)}} \in [-1,1]$	h_t 和 h_k 无关则值为 0、正相关值为正、负相关值为负
Q-统计量 (Q-statistic)	$Q := \frac{ad-bc}{ad+bc}$	符号关系和相关系数相同
κ-统计量 (κ-statistic)	$\kappa := \frac{p_1-p_2}{1-p_2}$	若两分类器在 D 上完全一致 $\kappa=1$, 若仅是偶然一致 $\kappa=0$

著缺陷 [17, 24] . 如何理解多样性, 被认为是集成学习中的圣杯问题.

多样性增强方法? 在现实任务中, 很难直接对公式 4 进行优化, 不仅由于它是定义在整个样本空间上, 还由于 $(h_t - H)^2$ 是一个不可直接操作的多样性度量, 它仅在集合构造好之后才能进行估计. 此外, 公式 4 的推导只适用于回归任务, 难以直接推广到分类任务中. 增强多样性的一般思路是引入随机性.

- **数据样本扰动**. 从给定数据集 D 中采样产生不同的数据子集 $\tilde{D}$, 再利用不同的数据子集训练不同的个体学习器. 例如 Bagging 使用自助采样, AdaBoost 使用序列采样. 对不稳定基学习器 (如决策树、神经网络) 很有效, 而稳定基学习器 (如线性学习器、支持向量机、朴素贝叶斯、k 近邻) 对此不敏感.
- **输入属性扰动**. 从初始的高维属性空间投影产生低维属性空间, 不同的子空间提供了观察数据的不同视角, 再利用不同的子空间训练不同的个体学习器. 该方法适用于包含大量冗余数据的属性, 否则则不适用.
- **算法参数扰动**. 例如改变神经网络的隐层神经元数、初始连接权值等. 负相关法 (negative correlation) [18] 显式使用正则化强制个体神经网络使用不同的参数. 对参数较少的算法, 可通过将其学习过程中某些环节用其他类似方式代替 (例如改变决策树属性选择机制), 从而达到扰动的目的.
- **输出表示扰动**. 例如, 翻转法 (flipping output) [4] 随机改变一些训练样本的标记, 输出调制法 (output smearing) [4] 将分类输出转化为回归输出后构建个体学习器. ECOC 法 [7] 将多分类任务拆解成一系列二分类任务来训练学习器.

1.4 本文内容

本文介绍两大类的集成学习方法, 如表 3 所示.

2 Bagging

Bagging 的基本思路? 一种朴素地增加基分类器多样性的方案是用训练集 D 的不同子集训练不同的个体学习器. 但是, 我们同时希望个体学习器不能太差. 如果每个子集互不相交, 则每个集学习器只用到了一小部分训练数据, 甚至不足以有效学习. 因此, Bagging (boostrap aggregating) [1] 从训练集 D 采样相互有交叠的采样子集.

2.1 自助法采样

给定大小为 m 的数据集 D, 自助法采样 (bootstrap sampling) [8] 从 D 中有放回地采样 m 次得到数据集 $\tilde{D}$.

引理 3. *D 中的某个样本在自助法采样中不被采到的概率约是 36.8%. 也就是说, 在 D 中大约有 36.8% 的样本未出现在采样数据集 $\tilde{D}$ 中.*

Proof. 在一次采样中, 该样本未被采到的概率是 $1-\frac{1}{m}$. 取极限得到 $\lim_{m\to\infty}\left(1-\frac{1}{m}\right)^m = \frac{1}{\mathrm{e}} \approx 36.8\%$. □

自助法采样用来划分验证集的优缺点? 给定数据集 D, 我们可以把自助法采样得到的数据集 $\tilde{D}$ 作为训练集, 用 $D-\tilde{D}$ 作为验证集, 这样的验证结果称为包外验证 (out-of-bag estimation) [1, 27]. 自助法在数据集较小, 难以划分训练/验证集时很有用, 训练集大小仍是 m. 然而, 自助法产生的数据集 $\tilde{D}$ 改变了初始数据集分布, 会引入估计偏差. 因此, 当初始数据量足够时, 交叉验证和留出法更常用.

Table 3: 两大类集成学习方法对比.

方法	个体学习器之间依赖关系	生成方式	作用	适用情形
Bagging	不存在强依赖	并行生成	降低方差	适用于易受样本扰动的学习器 (如不剪枝的决策树、神经网络等)
Boosting	存在强依赖	串行生成	降低偏差	适用于泛化性能相当弱的个体学习器

2.2 Bagging 的具体过程

Bagging 的具体过程包括如下三步.

1. 从训练集 D 中由自助法采样得到 T 个采样集 $\tilde{D}_1, \tilde{D}_2, \ldots, \tilde{D}_T$.
2. 基于自助采样集 $\tilde{D}_t$ 训练基学习器 h_t.
3. 使用相对多数投票 (分类任务) 或简单平均 (回归任务) 得到 H.

在利用 Bagging 进行包外估计时, 我们考虑那些未使用 $\boldsymbol{x}$ 训练的基学习器在 $\boldsymbol{x}$ 上的预测

$$H_{\text{oob}}(\boldsymbol{x}) := \arg\max_{c} \sum_{t=1}^{T} \mathbb{I}(\boldsymbol{x} \notin \tilde{D}_t)\, \mathbb{I}(h_t(\boldsymbol{x}) = c)\,. \quad (9)$$

那么 Bagging 的包外误差估计为

$$E_{\text{oob}} := \frac{1}{m} \sum_{i=1}^{m} I(H_{\text{oob}}(\boldsymbol{x}_i) \neq y_i)\,. \quad (10)$$

在下一章, 我们将看到一个 Bagging 的一个扩展变体: 随机森林.

3 Boosting

Boosting 的基本思路? 先从初始训练集 D 训练出一个基学习器, 再根据基学习器的表现对训练样本分布进行调整. 使得先前做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, 如此重复进行.

Boosting 源于 [22] 对 [14] 提出的 "弱学习器是否等价于强学习器" 这个重要理论问题的构造性证明. 最初的 Boosting 算法仅有理论意义, 经数年努力后, [9] 提出 AdaBoost, 并因此获得理论计算机科学方面的重要奖项— Gödel (哥德尔) 奖.

在本节, 我们只考虑二分类问题 $y \in \{-1, 1\}$.

3.1 AdaBoost

AdaBoost 的基本思路? AdaBoost (adaptive boosting) 通过给样本加权实现来对训练样本的分布进行调整. 原来的优化目标是想最小化错误率

$$e(h) := \frac{1}{m} \sum_{i=1}^{m} \mathbb{I}(h(\boldsymbol{x}_i) \neq y_i)\,, \quad (11)$$

AdaBoost 想要最小化分布调整 (样本加权) 后的错误率

$$\tilde{e}(h_t, \boldsymbol{w}_t) := \frac{\sum_{i=1}^{m} w_{t,i}\, \mathbb{I}(h_t(\boldsymbol{x}_i) \neq y_i)}{\sum_{i=1}^{m} w_{t,i}}\,. \quad (12)$$

其中 $w_{t,i}$ 是在第 t 轮 (对应第 t 个基学习器) 第 i 个样本的权重. 因此, AdaBoost 的关键是如何设计样本权重.

如何设计样本权重 $\boldsymbol{w}_{t+1}$? $\boldsymbol{w}_{t+1}$ 设计原则是使 h_{t+1} 更多关注 h_t 做错的样本. 因此, 我们调节 $\boldsymbol{w}_{t+1}$ 使得 $\tilde{e}(h_t, \boldsymbol{w}_{t+1})$ 尽可能小, 即尽可能接近 0.5.

定理 4. 第 $t+1$ 轮的样本权重的更新规则是

$$w_{t+1,i} := \begin{cases} w_{t,i} \sqrt{\dfrac{1-\tilde{e}(h_t, \boldsymbol{w}_t)}{\tilde{e}(h_t, \boldsymbol{w}_t)}} & \text{若 } h_t(\boldsymbol{x}_i) \neq y_i\,; \\ w_{t,i} \sqrt{\dfrac{\tilde{e}(h_t, \boldsymbol{w}_t)}{1-\tilde{e}(h_t, \boldsymbol{w}_t)}} & \text{若 } h_t(\boldsymbol{x}_i) = y_i\,. \end{cases} \quad (13)$$

Proof. 在第 t 轮, 我们有

$$\tilde{e}(h_t, \boldsymbol{w}_t)(1-\tilde{e}(h_t, \boldsymbol{w}_t)) = (1-\tilde{e}(h_t, \boldsymbol{w}_t))\tilde{e}(h_t, \boldsymbol{w}_t)\,. \quad (14)$$

这等价于

$$\tilde{e}(h_t, \boldsymbol{w}_t) \sqrt{\frac{1-\tilde{e}(h_t, \boldsymbol{w}_t)}{\tilde{e}(h_t, \boldsymbol{w}_t)}} = (1-\tilde{e}(h_t, \boldsymbol{w}_t)) \sqrt{\frac{\tilde{e}(h_t, \boldsymbol{w}_t)}{1-\tilde{e}(h_t, \boldsymbol{w}_t)}}\,. \quad (15)$$

等价于

$$\begin{aligned} &\sum_{i=1}^{m} w_{t,i} \sqrt{\frac{1-\tilde{e}(h_t, \boldsymbol{w}_t)}{\tilde{e}(h_t, \boldsymbol{w}_t)}}\, \mathbb{I}(h_t(\boldsymbol{x}_i) \neq y_i) \\ =& \sum_{i=1}^{m} w_{t,i} \sqrt{\frac{\tilde{e}(h_t, \boldsymbol{w}_t)}{1-\tilde{e}(h_t, \boldsymbol{w}_t)}}\, \mathbb{I}(h_t(\boldsymbol{x}_i) = y_i)\,. \end{aligned} \quad (16)$$

通过令公式 13 , 可以得到我们的目标

$$\sum_{i=1}^{m} w_{t+1,i}\,\mathbb{I}(h_t(\boldsymbol{x}_i)\neq y_i)=\sum_{i=1}^{m} w_{t+1,i}\,\mathbb{I}(h_t(\boldsymbol{x}_i)= y_i)\,, \tag{17}$$

即 $\tilde{e}(h_t,\boldsymbol{w}_{t+1})=0.5$ □

AdaBoost 的基学习器结合策略? 由于不同基学习器的准确率不同, AdaBoost 使用加权平均对不同基学习器进行结合.

$$H(\boldsymbol{x}):=\sum_{t=1}^{T}\alpha_t h(\boldsymbol{x})_t\,, \tag{18}$$

并使 α_t 和准确率成正比, 由于 $\sqrt{\frac{1-\tilde{e}(h_t,\boldsymbol{w}_t)}{\tilde{e}(h_t,\boldsymbol{w}_t)}}\in[1,\infty)$, 我们令

$$\alpha_t:=\log\sqrt{\frac{1-\tilde{e}(h_t,\boldsymbol{w}_t)}{\tilde{e}(h_t,\boldsymbol{w}_t)}}\in[0,\infty) \tag{19}$$

完整的 AdaBoost 学习算法见算法 1 . 其中, 若 h_t 劣于随机猜测, 则抛弃 h_t, 学习过程停止. 在此种情况下, 初始设置的学习轮数 T 也许还远未达到, 可能导致最终集成中只包含很少的基学习器而性能不佳.

Algorithm 1 AdaBoost 学习算法.

$\boldsymbol{w}_1\leftarrow\frac{1}{m}\mathbf{1}$ ▷ 初始化样本权重分布
for $t\leftarrow 1$ **to** T **do**
  基于 $\boldsymbol{w}_t$ 加权的数据训练学习器 h_t
  $\tilde{e}(h_t,\boldsymbol{w}_t)\leftarrow\frac{\sum_{i=1}^{m} w_{t,i}\,\mathbb{I}(h_t(\boldsymbol{x}_i)\neq y_i)}{\sum_{i=1}^{m} w_{t,i}}$ ▷ h_t 的加权误差
  if $\tilde{e}(h_t,\boldsymbol{w}_t)>0.5$ **then**
    break ▷ 检查 h_t 是否比随机猜测好
  $\alpha_t\leftarrow\frac{1}{2}\log\frac{1-\tilde{e}(h_t,\boldsymbol{w}_t)}{\tilde{e}(h_t,\boldsymbol{w}_t)}$ ▷ h_t 的权重
  for $i\leftarrow 1$ **to** m **do**
    $$w_{t+1,i}\leftarrow\begin{cases} w_{t,i}\sqrt{\frac{1-\tilde{e}(h_t,\boldsymbol{w}_t)}{\tilde{e}(h_t,\boldsymbol{w}_t)}} & \text{若 } h_t(\boldsymbol{x}_i)\neq y_i\,; \\ w_{t,i}\sqrt{\frac{\tilde{e}(h_t,\boldsymbol{w}_t)}{1-\tilde{e}(h_t,\boldsymbol{w}_t)}} & \text{若 } h_t(\boldsymbol{x}_i)= y_i \end{cases}$$
return $\operatorname{sign}\sum_{t=1}^{T}\alpha_t h(\boldsymbol{x})_t$

如何对加权的数据进行学习? 算法 1 实际上利用重赋权对数据进行加权. 如果基学习器无法接受带权样本, 我们可以通过重采样 (re-sampling) 来处理, 即在每一轮学习中, 根据样本权重 $\boldsymbol{w}_t$ 对训练集 D 重新进行采样, 再用重采样得到的样本集对基学习器进行训练.

重赋权和重采样方法的优劣比较? 重赋权和重采样两种方法没有显著的优劣差别. 不过, 当 h_t 不满足基本条件时, 若采用重采样法, 可根据当前分布重新对训练样本进行采样, 再基于新的采样结果重新训练出集学习器, 从而使得学习过程可以持续到预设的 T 轮完成.

3.2 AdaBoost 的统计视角

引理 5. *更新规则*

$$w_{t+1,i}:=\begin{cases} w_{t,i}\sqrt{\frac{1-\tilde{e}(h_t,\boldsymbol{w}_t)}{\tilde{e}(h_t,\boldsymbol{w}_t)}} & \text{若 } h_t(\boldsymbol{x}_i)\neq y_i\,; \\ w_{t,i}\sqrt{\frac{\tilde{e}(h_t,\boldsymbol{w}_t)}{1-\tilde{e}(h_t,\boldsymbol{w}_t)}} & \text{若 } h_t(\boldsymbol{x}_i)= y_i \end{cases} \tag{20}$$

等价于

$$w_{t+1,i}=w_{t,i}\exp(-y_i\alpha_t h_t(\boldsymbol{x}_i))\,. \tag{21}$$

Proof. 由于当 $h_t(\boldsymbol{x}_i)\neq y_i$ 时 $y_i h_t(\boldsymbol{x}_i)=-1$, 当 $h_t(\boldsymbol{x}_i)=y_i$ 时 $y_i h_t(\boldsymbol{x}_i)=1$,

$$\begin{aligned} w_{t+1,i}&=w_{t,i}\left(\frac{1-\tilde{e}(h_t,\boldsymbol{w}_t)}{\tilde{e}(h_t,\boldsymbol{w}_t)}\right)^{-\frac{1}{2}y_i h_t(\boldsymbol{x}_i)} \\ &=w_{t,i}\exp\log\left(\frac{1-\tilde{e}(h_t,\boldsymbol{w}_t)}{\tilde{e}(h_t,\boldsymbol{w}_t)}\right)^{-\frac{1}{2}y_i h_t(\boldsymbol{x}_i)} \\ &=w_{t,i}\exp\left(-y_i h_t(\boldsymbol{x}_i)\log\left(\frac{1-\tilde{e}(h_t,\boldsymbol{w}_t)}{\tilde{e}(h_t,\boldsymbol{w}_t)}\right)^{\frac{1}{2}}\right) \\ &=w_{t,i}\exp(-y_i\alpha_t h_t(\boldsymbol{x}_i))\,. \end{aligned} \tag{22}$$

□

推论 6.

$$w_{t+1,i}=\frac{1}{m}\exp(-y_i H(\boldsymbol{x}_i))\,. \tag{23}$$

Proof. 递归利用引理 5 , 以及 $\boldsymbol{w}_1=\frac{1}{m}\mathbf{1}$. □

定理 7. *AdaBoost 的基学习器 h_{t+1} 最小化带权分布调整 (样本加权) 后的错误率可近似看作是基于加性模型以类似牛顿迭代法来优化指数损失函数 [10] .*

$$\mathcal{L}(\boldsymbol{\theta}):=\frac{1}{m}\sum_{i=1}^{m}\exp(-y_i H(\boldsymbol{x}_i))\,. \tag{24}$$

Proof. 根据推论 6 ,

$$
\begin{aligned}
H^\star &:= \arg\min_H \ \tilde{e}(h_{t+1}, \boldsymbol{w}_{t+1}) \\
&= \arg\min_H \ \frac{\sum_{i=1}^m w_{t+1,i}\, \mathbb{I}(h_{t+1}(\boldsymbol{x}_i) \neq y_i)}{\sum_{i=1}^m w_{t+1,i}} \\
&= \arg\min_H \ \sum_{i=1}^m w_{t+1,i}\, \mathbb{I}(h_{t+1}(\boldsymbol{x}_i) \neq y_i) \\
&\leq \arg\min_H \ \sum_{i=1}^m w_{t+1,i} \\
&= \arg\min_H \ \frac{1}{m} \sum_{i=1}^m \exp\left(-y_i \sum_{t=1}^T \alpha_t h_t(\boldsymbol{x}_i)\right) \\
&= \frac{1}{m} \sum_{i=1}^m \exp(-y_i H(\boldsymbol{x}_i)) \,. \qquad (25)
\end{aligned}
$$

□

三种常见的替代损失函数的比较? 对二分类问题 $y \in \{-1, 1\}$, 我们想要学习器 $h\colon \mathcal{X} \to \{-1, 1\}$ 优化如下 0/1 损失函数

$$
\begin{aligned}
h^\star &:= \arg\min_h \ \frac{1}{m} \sum_{i=1}^m \mathbb{I}(h(\boldsymbol{x}_i) \neq y_i) \\
&= \arg\min_h \ \frac{1}{m} \sum_{i=1}^m \mathbb{I}(y_i h(\boldsymbol{x}_i) < 0) \,. \qquad (26)
\end{aligned}
$$

但是, 0/1 损失函数非凸、不连续、数学性质不好, 使其不易于直接求解. 于是, 我们通常用其他一些函数, 称为替代损失 (surrogate loss), 来代替 0/1 损失函数. 替代损失函数一般具有较好的数学性质, 如它们通常是凸的、连续、是 0/1 损失的上界. 至此, 我们看到了三种常用的替代损失函数:

- 在对数几率回归中使用的对数几率损失.*

$$
\begin{aligned}
h^\star &:= \arg\min_h \ \frac{1}{m} \sum_{i=1}^m \log(1 + \exp(-y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b))) \\
&= \arg\min_h \ \frac{1}{m} \sum_{i=1}^m \log(1 + \exp(-y_i h(\boldsymbol{x}_i))) \,. \qquad (27)
\end{aligned}
$$

- 在支持向量机中使用的合页损失.

$$
\begin{aligned}
h^\star &:= \arg\min_h \ \frac{1}{m} \sum_{i=1}^m \max(1 - y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b), 0) \\
&= \arg\min_h \ \frac{1}{m} \sum_{i=1}^m \max(1 - y_i h(\boldsymbol{x}_i), 0) \,. \qquad (28)
\end{aligned}
$$

*这里使用了 $y \in \{-1, 1\}$ 版本的对数几率损失函数.

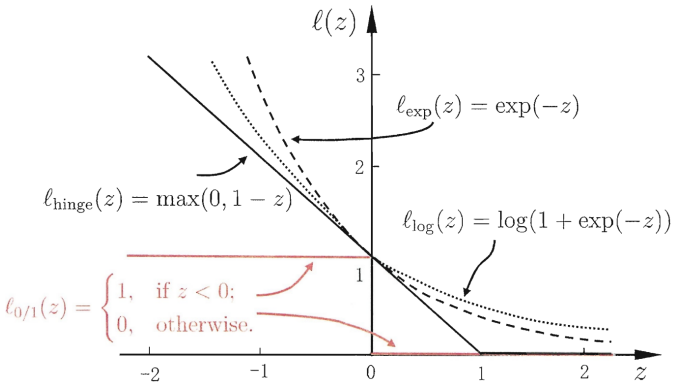


Figure 3: 三种常见的替代损失函数: 对数几率损失、合页损失、指数损失. 其中 $z := yh(\boldsymbol{x})$. 本图源于 [32] .

- 在 AdaBoost 中使用的指数损失.†

$$
h^\star := \arg\min_h \ \frac{1}{m} \sum_{i=1}^m \exp(-y_i h(\boldsymbol{x}_i)) \,. \qquad (29)
$$

这三种替代损失函数如图 3 所示.

AdaBoost 统计视角的局限? 统计视角产生的推论实际上和 AdaBoost 实际行为有相当大的差别 [19] . 以概率 $1 - \delta$, AdaBoost 的泛化误差界为 [9]

$$
e(H) \leq \hat{e}(H) + \tilde{\mathcal{O}}\left(\sqrt{\frac{\mathrm{VC}(h) \cdot T}{m}}\right) , \qquad (30)
$$

其中相比 $\mathcal{O}$ 记号, $\tilde{\mathcal{O}}$ 记号另外隐藏了对数项. 从该泛化误差界可以看出, 为了获得了更好的泛化性能, 基学习器的复杂度 ($\mathrm{VC}(h)$) 和学习轮数 (T) 应尽可能小. 但它不能解释为什么 AdaBoost 在训练误差达到零之后继续训练仍能提高泛化性能 [23]. 因此, 有不少学者认为, 统计视角本身虽很有意义, 但其阐释的是一个与 AdaBoost 相似的学习过程而非 AdaBoost 本身.

3.3 AdaBoost 的间隔视角

相比统计视角, 间隔视角 [23] 能直观地解释为什么 AdaBoost 能在训练误差达到零之后继续提高泛化性能.

定义 2 (间隔 (margin)). 二分类器 h 对某样本 $\boldsymbol{x}$ 的间隔为

$$
\gamma := yh(\boldsymbol{x}) \,. \qquad (31)
$$

†在这里为了符号统一, 将指数损失中的 H 写作 h.

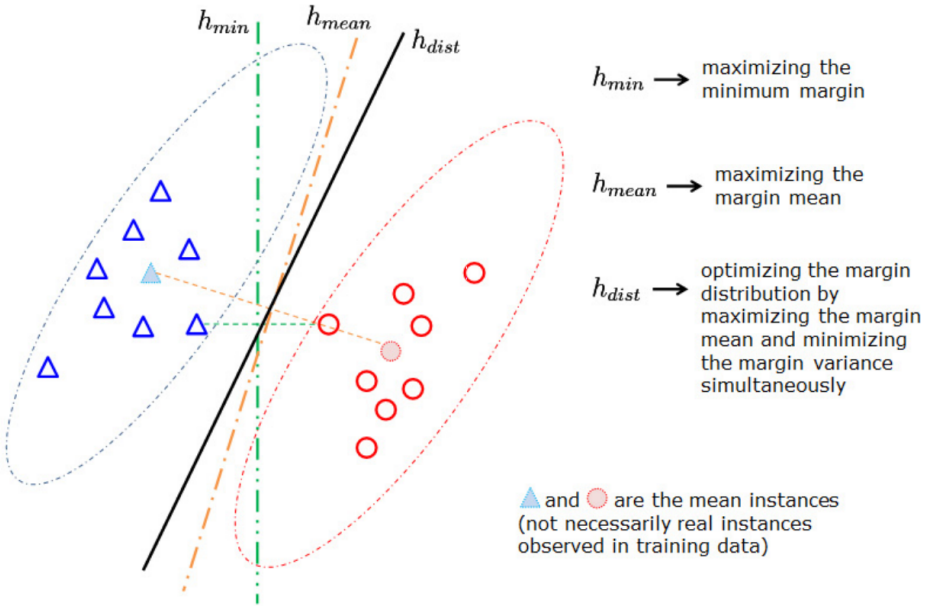


Figure 4: 优化最小间隔、间隔均值和间隔分布得到的分类边界图示. 本图源于 [31] .

集成的间隔为

$$\gamma := \sum_{t=1}^{T} \alpha_t y h_t(\boldsymbol{x}) = \sum_{t=1}^{T} \mathbb{I}(h_t(\boldsymbol{x}) = y)\alpha_t - \sum_{t=1}^{T} \mathbb{I}(h_t(\boldsymbol{x}) \neq y)\alpha_t \,. \tag{32}$$

以概率 $1-\delta$, 对任意间隔阈值 $\theta > 0$, AdaBoost 的泛化误差界为 [23]

$$e(H) \leq \Pr(yH(\boldsymbol{x}) \leq \theta) + \tilde{\mathcal{O}}\left(\sqrt{\frac{\mathrm{VC}(h)}{m\theta^2} + \log\frac{1}{\delta}}\right) \tag{33}$$

可以看出, 在其他影响因素固定时, 当训练集的间隔越大, 泛化误差越小. 因此, [23] 认为在训练误差达到零之后, AdaBoost 仍能通过增加间隔来减小泛化误差.

从公式 33 可以看出, 泛化误差界严重依赖于训练集中最小的间隔. 当最小的间隔很大时, $\Pr(yH(\boldsymbol{x}) \leq \theta)$ 会很大. 然而, 有理论和实验研究结果指出 [12, 21] , 优化间隔的分布比优化最小间隔更重要. 这个发现也对设计大间隔学习算法 (以支持向量机为代表) 具有启示意义, 例如相比支持向量机优化最小间隔, 可以同时最大化间隔的均值和最小化间隔的方差 [30] , 如图 4 所示.

3.4 GradientBoosting

受 AdaBoost 的统计视角的启发, 通过将迭代优化过程换为其他方法, 产生了多种变体 Boosting 算法. 本节介绍其中的一个代表, GradientBoosting [11] . GradientBoosting 考虑基于加性模型优化任意损失函数

$$\min_{\boldsymbol{\alpha}} \min_{H} \frac{1}{m}\sum_{i=1}^{m} \ell\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}_i), y_i\right). \tag{34}$$

GradientBoosting 的基本思路. 直接优化公式 34 十分复杂, GradientBoosting 采用前向分步算法 (forward stagewise algorithm) 求解. 每次只学习一个基学习器 h_t 及其权重 α_t, 逐步最小化损失函数. 使用下标 T 显式表示集成是由 T 个基学习器得到

$$H_T(\boldsymbol{x}) := \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}) \,. \tag{35}$$

用前向分步算法每步只需优化如下目标

$$\min_{\alpha_t} \min_{h_t} \frac{1}{m}\sum_{i=1}^{m} \ell(H_{t-1}(\boldsymbol{x}_i) + \alpha_t h_t(\boldsymbol{x}_i), y_i) \,. \tag{36}$$

定理 8. *对公式 36 内层的优化问题, h_t 的最优值是对任意 $i = 1, 2, \ldots, m$,*

$$h_t(\boldsymbol{x}_i) \approx -\frac{\partial \ell}{\partial H_{t-1}(\boldsymbol{x})}\bigg|_{\boldsymbol{x}=\boldsymbol{x}_i} . \tag{37}$$

Proof. 类似于梯度下降 (参见第 4 章), GradientBoost-

Table 4: 梯度下降和 GradientBoosting 对比.

	梯度下降	GradientBoosting
更新形式	$\boldsymbol{\theta}_t := \boldsymbol{\theta}_{t-1} + \eta_t \Delta\boldsymbol{\theta}_t$	$H_t(\boldsymbol{x}) := H_{t-1}(\boldsymbol{x}) + \alpha_t h_t(\boldsymbol{x})$
内层变量最优值	$\Delta\boldsymbol{\theta}_t := -\frac{\partial \ell}{\partial \boldsymbol{\theta}}\Big\|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{t-1}}$	$h_t(\boldsymbol{x}_i) \approx -\frac{\partial \ell}{\partial H_{t-1}(\boldsymbol{x})}\Big\|_{\boldsymbol{x}=\boldsymbol{x}_i}$
外层变量最优值	$\eta_t := \arg\min_{\eta_t} \ell(\boldsymbol{\theta}_{t-1} + \eta_t \Delta\boldsymbol{\theta}_t)$ 或设为很小的常数	$\alpha_t := \arg\min_{\alpha_t} \frac{1}{m}\sum_{i=1}^m \ell(H_{t-1}(\boldsymbol{x}_i) + \alpha_t h_t(\boldsymbol{x}_i), y_i)$
累积形式	$\boldsymbol{\theta}_T = \sum_{t=1}^T \eta_t \Delta\boldsymbol{\theta}_t$	$H_T(\boldsymbol{x}) = \sum_{t=1}^T \alpha_t h_t(\boldsymbol{x})$

Table 5: 牛顿法和 XGBoost 对比.

	牛顿法	XGBoost
更新形式	$\boldsymbol{\theta}_t := \boldsymbol{\theta}_{t-1} + \Delta\boldsymbol{\theta}_t$	$H_t(\boldsymbol{x}) := H_{t-1}(\boldsymbol{x}) + h_t(\boldsymbol{x})$
变量最优值	$\Delta\boldsymbol{\theta}_t := -\left(\frac{\partial^2 \ell}{\partial \boldsymbol{\theta}^2}\Big\|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{t-1}}\right)^{-1} \frac{\partial \ell}{\partial \boldsymbol{\theta}}\Big\|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{t-1}}$	$h_t(\boldsymbol{x}_i) \approx -\left(\frac{\partial^2 \ell}{\partial H_{t-1}(\boldsymbol{x})^2}\Big\|_{\boldsymbol{x}=\boldsymbol{x}_i}\right)^{-1} \frac{\partial \ell}{\partial H_{t-1}(\boldsymbol{x})}\Big\|_{\boldsymbol{x}=\boldsymbol{x}_i}$
累积形式	$\boldsymbol{\theta}_T = \sum_{t=1}^T \Delta\boldsymbol{\theta}_t$	$H_T(\boldsymbol{x}) = \sum_{t=1}^T h_t(\boldsymbol{x})$

ing 对损失函数进行一阶泰勒展开

$$\begin{aligned} h_t &:= \arg\min_{h_t} \frac{1}{m}\sum_{i=1}^m \ell(H_{t-1}(\boldsymbol{x}_i) + \alpha_t h_t(\boldsymbol{x}_i), y_i) \\ &= \arg\min_{h_t} \frac{1}{m}\sum_{i=1}^m \Big(\ell(H_{t-1}(\boldsymbol{x}_i), y_i) \\ &\qquad + \alpha_t h_t(\boldsymbol{x}_i)\frac{\partial \ell}{\partial H_{t-1}(\boldsymbol{x})}\Big|_{\boldsymbol{x}=\boldsymbol{x}_i}\Big) \\ &= \arg\min_{h_t} \sum_{i=1}^m h_t(\boldsymbol{x}_i)\frac{\partial \ell}{\partial H_{t-1}(\boldsymbol{x})}\Big|_{\boldsymbol{x}=\boldsymbol{x}_i}. \end{aligned} \tag{38}$$

因此, h_t 的最优值是使 $h_t(\boldsymbol{x}_i)$ 近似 $-\frac{\partial \ell}{\partial H_{t-1}(\boldsymbol{x})}\Big|_{\boldsymbol{x}=\boldsymbol{x}_i}$. □

梯度下降和 GradientBoosting 的对比如表 4 所示. 完整的 GradientBoosting 学习算法见算法 2 . 在下一章, 我们将看到一个 GradientBoosting 的一个扩展变体: GBDT.

Algorithm 2 GradientBoosting 学习算法.

$H_0 \leftarrow \arg\min_c \frac{1}{m}\sum_{i=1}^m \ell(c, y_i)$ ▷ 初始化集成结果
for $t \leftarrow 1$ **to** T **do**
  for $i \leftarrow 1$ **to** m **do**
    $r_i \leftarrow -\frac{\partial \ell}{\partial H_{t-1}(\boldsymbol{x})}\Big|_{\boldsymbol{x}=\boldsymbol{x}_i}$
  基于训练数据 $D_t := \{(\boldsymbol{x}_i, r_i)\}_{i=1}^m$ 训练学习器 h_t.
  $\alpha_t \leftarrow \arg\min_{\alpha_t} \frac{1}{m}\sum_{i=1}^m \ell(H_{t-1}(\boldsymbol{x}_i) + \alpha_t h_t(\boldsymbol{x}_i), y_i)$
  $H_t(\boldsymbol{x}) \leftarrow H_{t-1}(\boldsymbol{x}) + \alpha_t h_t(\boldsymbol{x})$
return $H_T(\boldsymbol{x})$

GradientBoosting 随后得到了许多扩展. 例如, 相比 GradientBoosting 对损失函数进行一阶泰勒近似, XGBoost (extreme gradient boosting) [5] 使用损失函数的二阶泰勒近似.

定理 9. 对 *XGBoost*, h_t 的最优值是对任意 $i = 1, 2, \ldots, m$,

$$h_t(\boldsymbol{x}_i) \approx -\left(\frac{\partial^2 \ell}{\partial H_{t-1}(\boldsymbol{x})^2}\Big|_{\boldsymbol{x}=\boldsymbol{x}_i}\right)^{-1} \frac{\partial \ell}{\partial H_{t-1}(\boldsymbol{x})}\Big|_{\boldsymbol{x}=\boldsymbol{x}_i}. \tag{39}$$

Proof. 类似于牛顿法 (参见第 4 章), XGBoost 对损失函数进行二阶泰勒展开

$$\begin{aligned} h_t &:= \arg\min_{h_t} \frac{1}{m}\sum_{i=1}^m \ell(H_{t-1}(\boldsymbol{x}_i) + h_t(\boldsymbol{x}_i), y_i) \\ &= \arg\min_{h_t} \frac{1}{m}\sum_{i=1}^m \Bigg(\ell(H_{t-1}(\boldsymbol{x}_i), y_i) \\ &\qquad + h_t(\boldsymbol{x}_i)\frac{\partial \ell}{\partial H_{t-1}(\boldsymbol{x})}\Big|_{\boldsymbol{x}=\boldsymbol{x}_i} \\ &\qquad + \frac{1}{2}h_t(\boldsymbol{x}_i)^2\left(\frac{\partial^2 \ell}{\partial H_{t-1}(\boldsymbol{x})^2}\Big|_{\boldsymbol{x}=\boldsymbol{x}_i}\right)^{-1}\Bigg) \\ &= \arg\min_{h_t} \sum_{i=1}^m \Bigg(h_t(\boldsymbol{x}_i)\frac{\partial \ell}{\partial H_{t-1}(\boldsymbol{x})}\Big|_{\boldsymbol{x}=\boldsymbol{x}_i} \\ &\qquad + \frac{1}{2}h_t(\boldsymbol{x}_i)^2\left(\frac{\partial^2 \ell}{\partial H_{t-1}(\boldsymbol{x})^2}\Big|_{\boldsymbol{x}=\boldsymbol{x}_i}\right)^{-1}\Bigg) \end{aligned} \tag{40}$$

通过对 $h_t(\boldsymbol{x}_i)$ 求偏导为零, 可得 h_t 的最优值. □

牛顿法和 XGBoost 的对比如表 5 所示.

4 快问快答

Bagging 和 *Boosting* 的区别?
答案见上文.

References

[1] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996. 4

[2] L. Breiman. Stacked regressions. *Machine Learning*, 24(1):49–64, 1996. 2

[3] L. Breiman. Stacked regressions. *Machine Learning*, 24(1):49–64, 1996. 1

[4] L. Breiman. Randomizing outputs to increase prediction accuracy. *Machine Learning*, 40(3):229–242, 2000. 4

[5] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 785–794, 2016. 9

[6] T. G. Dietterich. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems (MCS)*, pages 1–15, 2000. 1

[7] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995. 4

[8] B. Efron and R. J. Tibshirani. *An introduction to the bootstrap*. Chapman & Hall, 1994. 4

[9] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997. 5, 7

[10] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting (with discussion). *Annals of Statistics*, 28(2):337–407, 2000. 6

[11] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001. 8

[12] W. Gao and Z. Zhou. On the doubt about margin explanation of boosting. *Artificial Intelligence*, 203:1–18, 2013. 8

[13] T. K. Ho, J. J. Hull, and S. N. Srihari. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):66–75, 1994. 2

[14] M. J. Kearns and L. G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 433–444, 1989. 5

[15] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998. 2

[16] A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. In *Advances in Neural Information Processing Systems 7 (NIPS)*, pages 231–238, 1994. 3

[17] L. I. Kuncheva and C. J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2):181–207, 2003. 4

[18] Y. Liu and X. Yao. Ensemble learning via negative correlation. *Neural Networks*, 12(10):1399–1404, 1999. 4

[19] D. Mease and A. J. Wyner. Evidence contrary to the statistical view of boosting. *Journal of Machine Learning Research*, 9:131–156, 2008. 7

[20] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 10(3):61–74, 1999. 2

[21] L. Reyzin and R. E. Schapire. How boosting the margin can also boost classifier complexity. In *Proceedings of the Twenty-Third International Conference on Machine Learning (ICML)*, pages 753–760, 2006. 8

[22] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990. 5

[23] R. E. Schapire, Y. Freund, P. Barlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML)*, pages 322–330, 1997. 7, 8

[24] E. K. Tang, P. N. Suganthan, and X. Yao. An analysis of diversity measures. *Machine Learning*, 65(1):247–271, 2006. 4

[25] K. M. Ting and I. H. Witten. Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10:271–289, 1999. 2

[26] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992. 2

[27] D. H. Wolpert and W. G. Macready. An efficient method to estimate bagging's generalization error. *Machine Learning*, 35(1):41–55, 1999. 4

[28] L. Xu, A. Krzyzak, and C. Y. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(3):418–435, 1992. 2

[29] B. Zadrozny and C. Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*, pages 609–616, 2001. 2

[30] T. Zhang and Z. Zhou. Optimal margin distribution machine. *CoRR*, abs/1604.03348, 2016. 8

[31] Z. Zhou. Large margin distribution learning. In *Proceedings of the Artificial Neural Networks in Pattern Recognition (ANNPR)*, pages 1–11, 2014. 8

[32] 周志华. 机器学习. 清华大学出版社, 2016. 3, 7