# 集成学习

汇报人：陈程

2024.9.15

# 目录

# 个体与集成

集成学习：通过构建并结合多个学习器来完成学习任务。

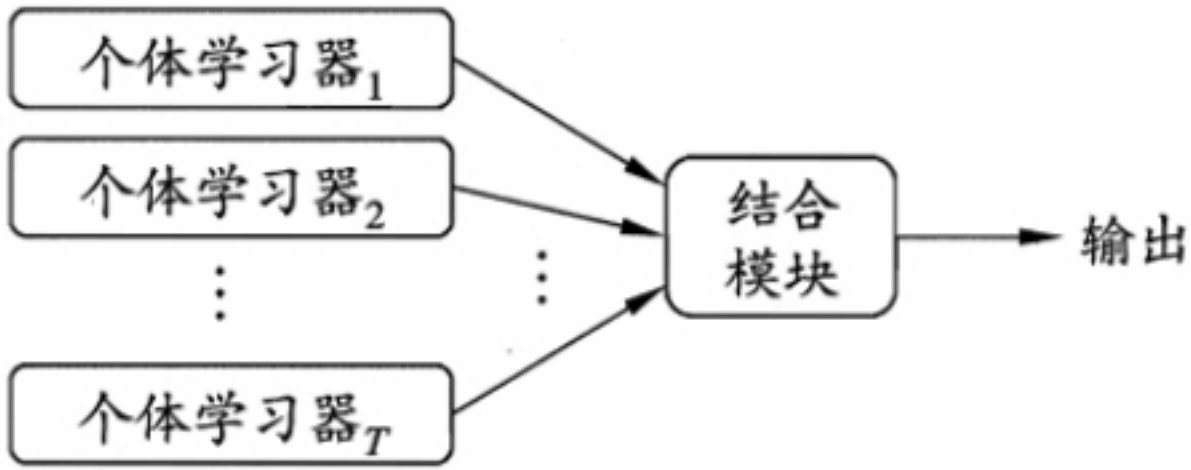



图 8.1 集成学习示意图

概念：同质集成、基学习器、异质集成、组件学习器

# 个体与集成

| | 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

**图 8.2** 集成个体应“好而不同”($h_i$ 表示第 $i$ 个分类器)

**个体分类器数目越多，集成的错误率越低。**

# Boosting

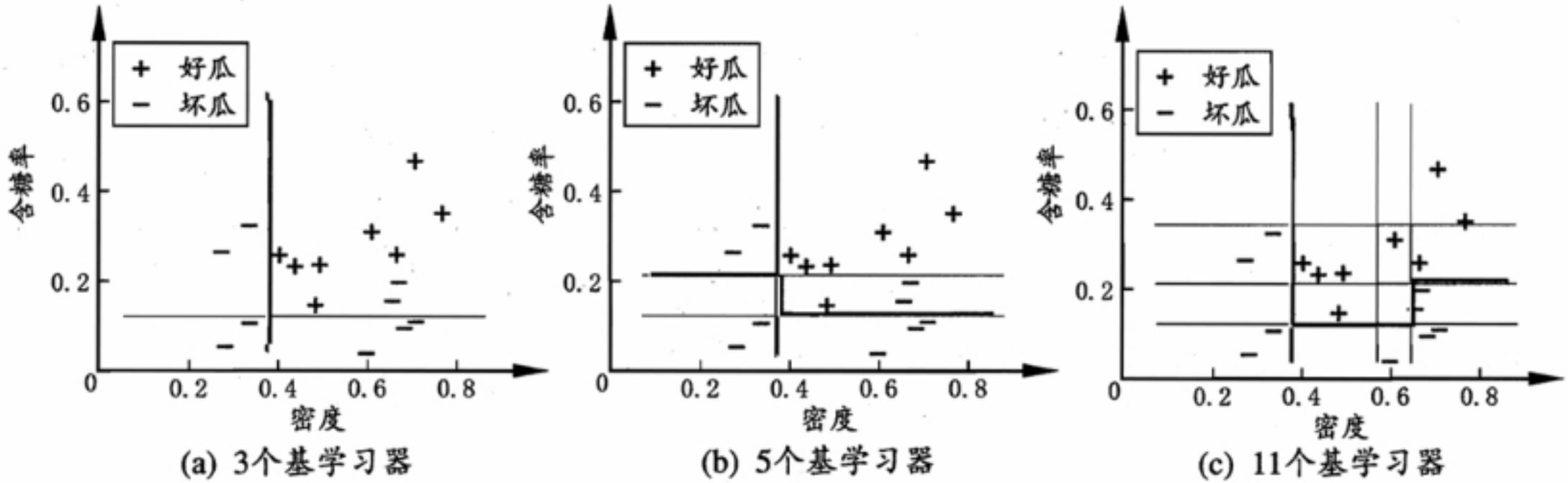
**工作机制**：先从初始训练集训练出一个基学习器，再根据基学习器的表现对训练样本分布进行调整，使得先前基学习器做错的训练样本在后续受到更多的关注，然后基于调整后的样本分布来训练下一个基学习器，如此重复进行，直至基学习器数目达到事先指定的值T，最终将这T个基学习器进行加权结合。

**代表：AdaBoost算法**

## 指数损失函数：

1. 先考虑指数损失函数$e^{-f(x)H(x)}$的含义：$f$为真实函数，对于样本$x$来说，$f(\boldsymbol{x}) \in \{+1,-1\}$只能取$+1$和$-1$，而$H(\boldsymbol{x})$是一个实数；当$H(\boldsymbol{x})$的符号与$f(x)$一致时，$f(\boldsymbol{x})H(\boldsymbol{x})>0$，因此$e^{-f(\boldsymbol{x})H(\boldsymbol{x})}=e^{-|H(\boldsymbol{x})|}<1$，且$|H(\boldsymbol{x})|$越大指数损失函数$e^{-f(\boldsymbol{x})H(\boldsymbol{x})}$越小（这很合理：此时$|H(\boldsymbol{x})|$越大意味着分类器本身对预测结果的信心越大，损失应该越小；若$|H(\boldsymbol{x})|$在零附近，虽然预测正确，但表示分类器本身对预测结果信心很小，损失应该较大）；当$H(\boldsymbol{x})$的符号与$f(\boldsymbol{x})$不一致时，$f(\boldsymbol{x})H(\boldsymbol{x})<0$，因此$e^{-f(\boldsymbol{x})H(\boldsymbol{x})}=e^{|H(\boldsymbol{x})|}>1$，且$|H(\boldsymbol{x})|$越大指数损失函数越大（这很合理：此时$|H(\boldsymbol{x})|$越大意味着分类器本身对预测结果的信心越大，但预测结果是错的，因此损失应该越大；若$|H(\boldsymbol{x})|$在零附近，虽然预测错误，但表示分类器本身对预测结果信心很小，虽然错了，损失应该较小）；

# AdaBoost




(a) 3个基学习器　(b) 5个基学习器　(c) 11个基学习器

**图 8.4** 西瓜数据集 3.0$\alpha$ 上 AdaBoost 集成规模为 3、5、11 时, 集成(红色)与基学习器(黑色)的分类边界.

# Bagging

工作原理: 基于自助采样法, 给定包含m个样本的数据集, 先随机取出一个样本放入采样集中，再把样本放回初始数据集，使得下次采样时该样本仍有可能被选中。采样出T个含m个训练样本的采样集, 然后基于每个采样集训练出一个基学习器，再将这些基学习器进行结合。在对预测输出进行结合时，Bagging通常对分类任务使用简单投票法，对回归任务使用简单平均法。

Bagging 与 AdaBoost 的不同:

1.Bagging是并行式集成学习方法，AdaBoost是串行式集成学习方法。

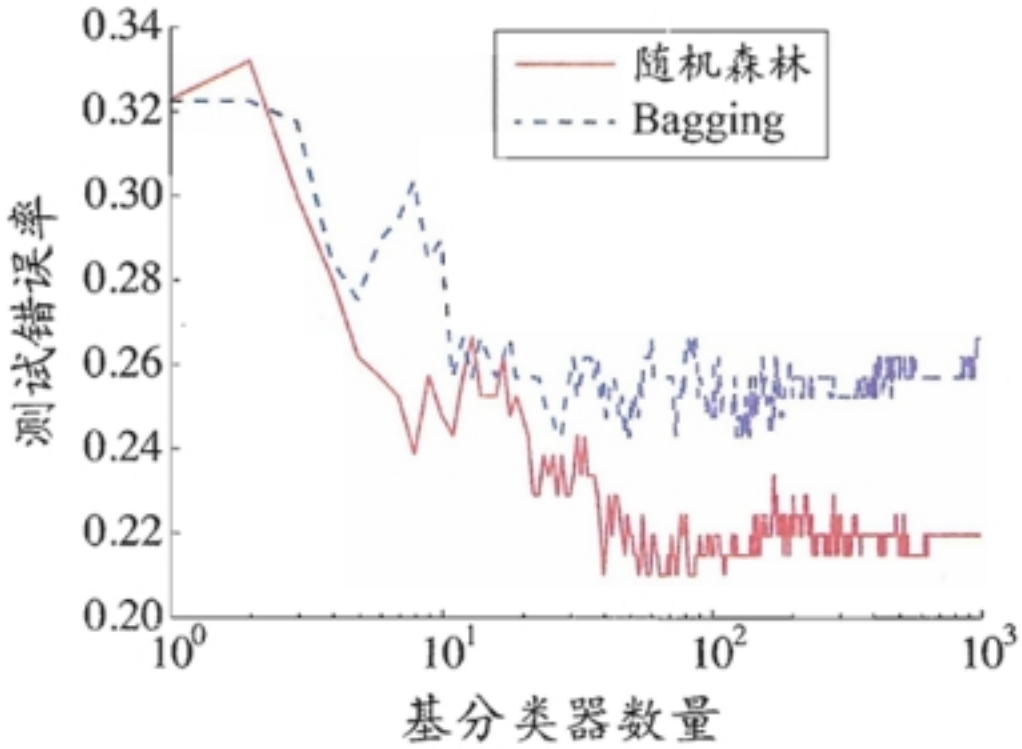
2.标准AdaBoost只适用于二分类任务，若要处理多分类或者回归任务，则需要进行修改。Bagging能够不经修改地用于多分类、回归任务。

# 随机森林

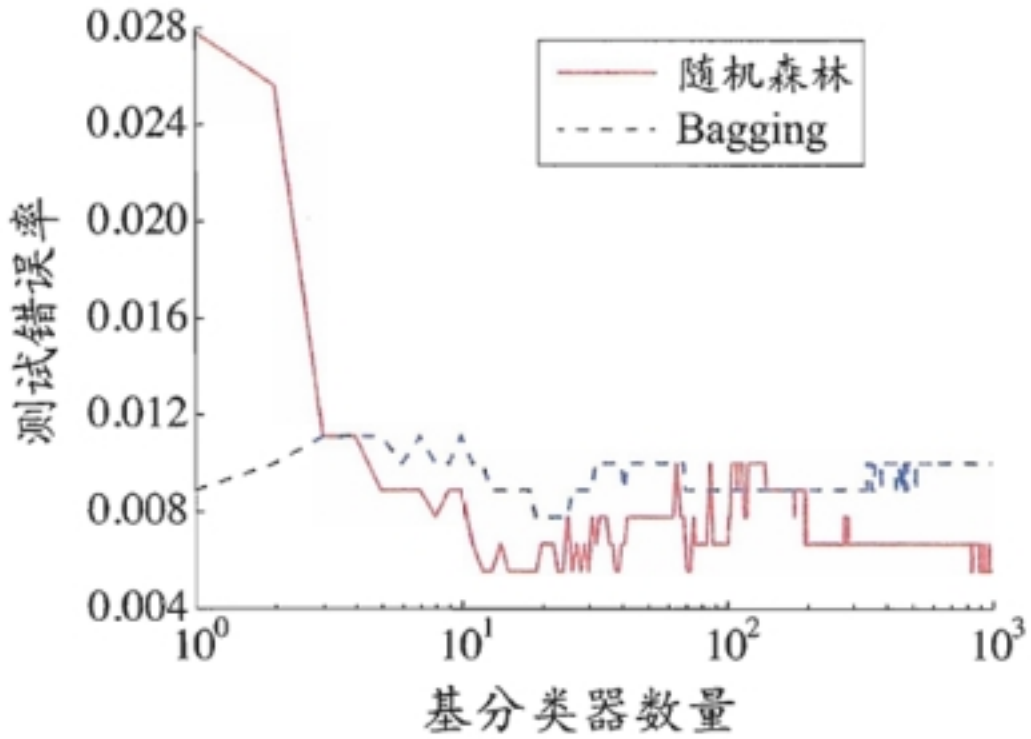
随机森林是Bagging的一个扩展变体。随机森林在以决策树为基学习器构建Bagging集成的基础上，进一步在决策树的训练过程中引入了随机属性选择。

优点：简单、容易实现、计算开销小。

与Bagging相比，Bagging中的基学习器的“多样性”仅通过样本扰动，随机森林中的基学习器的“多样性”不仅来自样本扰动，还来自属性扰动，使得最终集成的泛化性能可通过个体学习器之间的差异度的增加而进一步的提升。




(a) glass 数据集




(b) auto-mpg 数据集

**图 8.7** 在两个 UCI 数据上, 集成规模对随机森林与 Bagging 的影响

# 结合策略

假定集成包含 $T$ 个基学习器 $\{h_1, h_2, \ldots, h_T\}$, 其中 $h_i$ 在示例 $\boldsymbol{x}$ 上的输出为 $h_i(\boldsymbol{x})$. 本节介绍几种对 $h_i$ 进行结合的常见策略.

1.平均法
2.投票法
3.学习法

# 平均法

- 简单平均法(simple averaging)

$$H(\boldsymbol{x}) = \frac{1}{T}\sum_{i=1}^{T} h_i(\boldsymbol{x}) \ . \tag{8.22}$$

- 加权平均法(weighted averaging)

$$H(\boldsymbol{x}) = \sum_{i=1}^{T} w_i h_i(\boldsymbol{x}) \ . \tag{8.23}$$

其中 $w_i$ 是个体学习器 $h_i$ 的权重, 通常要求 $w_i \geqslant 0, \sum_{i=1}^{T} w_i = 1.$

选择: 在个体学习器性能相差较大时宜使用加权平均法, 而在个体学习器性能相近时宜使用简单平均法。

# 投票法

对分类任务来说, 学习器 $h_i$ 将从类别标记集合 $\{c_1, c_2, \ldots, c_N\}$ 中预测出一个标记, 最常见的结合策略是使用投票法(voting). 为便于讨论, 我们将 $h_i$ 在样本 $\boldsymbol{x}$ 上的预测输出表示为一个 $N$ 维向量 $(h_i^1(\boldsymbol{x}); h_i^2(\boldsymbol{x}); \ldots; h_i^N(\boldsymbol{x}))$, 其中 $h_i^j(\boldsymbol{x})$ 是 $h_i$ 在类别标记 $c_j$ 上的输出.

- 绝对多数投票法(majority voting)

$$H(\boldsymbol{x}) = \begin{cases} c_j, & \text{if } \sum_{i=1}^{T} h_i^j(\boldsymbol{x}) > 0.5 \sum_{k=1}^{N} \sum_{i=1}^{T} h_i^k(\boldsymbol{x}); \\ \text{reject}, & \text{otherwise.} \end{cases} \tag{8.24}$$

即若某标记得票过半数, 则预测为该标记; 否则拒绝预测.

- 相对多数投票法(plurality voting)

$$H(\boldsymbol{x}) = c_{\arg\max \sum_{i=1}^{T} h_i^j(\boldsymbol{x})}. \tag{8.25}$$

即预测为得票最多的标记, 若同时有多个标记获最高票, 则从中随机选取一个.

区别

- 加权投票法(weighted voting)

$$H(\boldsymbol{x}) = c_{\underset{j}{\arg\max} \sum_{i=1}^{T} w_i h_i^j(\boldsymbol{x})} . \tag{8.26}$$

与加权平均法类似, $w_i$ 是 $h_i$ 的权重, 通常 $w_i \geqslant 0$, $\sum_{i=1}^{T} w_i = 1$.

# 学习法

通过另一个学习器来进行结合。

Stacking:先从初始数据集训练出初级学习器，然后生成一个新数据集用于训练次级学习器。(个体学习器称作初级学习器，用于结合的学习器称为次级学习器)

# 课后题

试析随机森林为何比决策树 Bagging 集成的训练速度更快.

1.**随机属性选择**：随机森林在构建每棵决策树时，不仅有样本扰动，还有属性扰动，随机选择一部分特征作为候选特征，这种随机性有助于增加基学习器之间的差异性，从而提高集成模型的泛化性能。
2.**计算开销**: 随机森林在每个节点只考虑部分特征，这可能会减少计算量，而传统的决策树Bagging在每个节点分裂时，通常会考虑所有的特征，这意味着在特征数量较多的时候，决策树Bagging集成的计算量可能会相对较大，也就更耗时间。

AdaBoost 算法有多种推导方式, 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

$$H(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}) \tag{8.4}$$

$\alpha_t$：权重
$h_t$：第t个基学习器

来最小化指数损失函数(exponential loss function) [Friedman et al., 2000]

$$\ell_{\exp}(H \mid \mathcal{D}) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H(\boldsymbol{x})}] . \tag{8.5}$$

$\Rightarrow \sum_{x\in D} D(x) e^{-f(x)H(x)}$

$e^{-f(x)H(x)}$：指数损失函数

$Ex\sim D[\ ]$：表示在数据集D上以概率$D(x)$进行加权后的一个期望.

若 $H(\boldsymbol{x})$ 能令指数损失函数最小化, 则考虑式(8.5)对 $H(\boldsymbol{x})$ 的偏导

$$\frac{\partial \ell_{\exp}(H \mid \mathcal{D})}{\partial H(\boldsymbol{x})} = -e^{-H(\boldsymbol{x})} P(f(\boldsymbol{x}) = 1 \mid \boldsymbol{x}) + e^{H(\boldsymbol{x})} P(f(\boldsymbol{x}) = -1 \mid \boldsymbol{x}) , \tag{8.6}$$

$\rightarrow L = \sum_{x\in D} D(x) e^{-f(x)\cdot H(x)}$

$\frac{\partial L}{\partial H(x)} = \sum_{i=1}^{|D|} D(x_i)\left(e^{-H(x_i)}\mathbb{I}(f(x_i)=1) + e^{H(x_i)}\mathbb{I}(f(x_i)=-1)\right)$

$= -e^{-H(x_i)} P(f(x_i)=1|x_i) + e^{H(x_i)} P(f(x_i)=-1|x_i)$

$\mathbb{I}(\cdot)$是指示函数，若·为真则取值为1，否则取值0

令式(8.6)为零可解得

$$H(\boldsymbol{x}) = \frac{1}{2} \ln \frac{P(f(x) = 1 \mid \boldsymbol{x})}{P(f(x) = -1 \mid \boldsymbol{x})} , \tag{8.7}$$

因此, 有

$$\begin{aligned} \operatorname{sign}\left(H\left(\boldsymbol{x}\right)\right) &= \operatorname{sign}\left(\frac{1}{2} \ln \frac{P(f(x) = 1 \mid \boldsymbol{x})}{P(f(x) = -1 \mid \boldsymbol{x})}\right) \\ &= \begin{cases} 1, & P(f(x) = 1 \mid \boldsymbol{x}) > P(f(x) = -1 \mid \boldsymbol{x}) \\ -1, & P(f(x) = 1 \mid \boldsymbol{x}) < P(f(x) = -1 \mid \boldsymbol{x}) \end{cases} \\ &= \underset{y\in\{-1,1\}}{\arg\max} P(f(x) = y \mid \boldsymbol{x}) , \end{aligned} \tag{8.8}$$

这意味着 $\operatorname{sign}\left(H(\boldsymbol{x})\right)$ 达到了贝叶斯最优错误率. 换言之, 若指数损失函数最小化, 则分类错误率也将最小化; 这说明指数损失函数是分类任务原本 0/1 损失函数的一致的(consistent)替代损失函数. 由于这个替代函数有更好的数学性质, 例如它是连续可微函数, 因此我们用它替代 0/1 损失函数作为优化目标.

在 AdaBoost算法 中, 第一个分类器 $h_1$ 是通过直接将基学习算法用于初始数据分布而得; 此后迭代地生成 $h_t$ 和 $\alpha_t$, 当基分类器 $h_t$ 基于分布 $\mathcal{D}_t$ 产生后, 该基分类器的权重 $\alpha_t$ 应使得 $\alpha_t h_t$ 最小化指数损失函数

$$
\begin{aligned}
\ell_{\exp}\left(\alpha_t h_t \mid \mathcal{D}_t\right) &= \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_t}\left[e^{-f(\boldsymbol{x}) \alpha_t h_t(\boldsymbol{x})}\right] \\
&= \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_t}\left[e^{-\alpha_t} \mathbb{I}\left(f\left(\boldsymbol{x}\right)=h_t\left(\boldsymbol{x}\right)\right)+e^{\alpha_t} \mathbb{I}\left(f\left(\boldsymbol{x}\right) \neq h_t\left(\boldsymbol{x}\right)\right)\right] \\
&= e^{-\alpha_t} P_{\boldsymbol{x} \sim \mathcal{D}_t}\left(f\left(\boldsymbol{x}\right)=h_t\left(\boldsymbol{x}\right)\right)+e^{\alpha_t} P_{\boldsymbol{x} \sim \mathcal{D}_t}\left(f\left(\boldsymbol{x}\right) \neq h_t\left(\boldsymbol{x}\right)\right) \\
&= e^{-\alpha_t}\left(1-\epsilon_t\right)+e^{\alpha_t} \epsilon_t \ , \qquad (8.9)
\end{aligned}
$$

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

$$
\frac{\partial \ell_{\exp}\left(\alpha_t h_t \mid \mathcal{D}_t\right)}{\partial \alpha_t} = -e^{-\alpha_t}\left(1-\epsilon_t\right)+e^{\alpha_t} \epsilon_t \ , \qquad (8.10)
$$

令式(8.10)为零可解得

$$
\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \ , \qquad (8.11)
$$

这恰是图 8.3 中算法第 6 行的分类器权重更新公式.

AdaBoost 算法在获得 $H_{t-1}$ 之后样本分布将进行调整, 使下一轮的基学习器 $h_t$ 能纠正 $H_{t-1}$ 的一些错误. 理想的 $h_t$ 能纠正 $H_{t-1}$ 的全部错误, 即最小化

$$
\begin{aligned}
\ell_{\exp}(H_{t-1}+h_t \mid \mathcal{D}) &= \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}[e^{-f(\boldsymbol{x})(H_{t-1}(\boldsymbol{x})+h_t(\boldsymbol{x}))}] \\
&= \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})} e^{-f(\boldsymbol{x})h_t(\boldsymbol{x})}] \ . \qquad (8.12)
\end{aligned}
$$

注意到 $f^2(\boldsymbol{x}) = h_t^2(\boldsymbol{x}) = 1$, 式(8.12)可使用 $e^{-f(\boldsymbol{x})h_t(\boldsymbol{x})}$ 的泰勒展式近似为

$$
\begin{aligned}
\ell_{\exp}(H_{t-1}+h_t \mid \mathcal{D}) &\simeq \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\left(1-f(\boldsymbol{x})h_t(\boldsymbol{x})+\frac{f^2(\boldsymbol{x})h_t^2(\boldsymbol{x})}{2}\right)\right] \\
&= \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\left(1-f(\boldsymbol{x})h_t(\boldsymbol{x})+\frac{1}{2}\right)\right] \ . \qquad (8.13)
\end{aligned}
$$

于是, 理想的基学习器

$$
\begin{aligned}
h_t(\boldsymbol{x}) &= \arg\min_h \ell_{\exp}(H_{t-1}+h \mid \mathcal{D}) \\
&= \arg\min_h \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\left(1-f(\boldsymbol{x})h(\boldsymbol{x})+\frac{1}{2}\right)\right] \\
&= \arg\max_h \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})} f(\boldsymbol{x})h(\boldsymbol{x})\right] \\
&= \arg\max_h \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}\left[\frac{e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}]} f(\boldsymbol{x})h(\boldsymbol{x})\right] , \qquad (8.14)
\end{aligned}
$$

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

$$\mathcal{D}_t(\boldsymbol{x}) = \frac{\mathcal{D}(\boldsymbol{x})e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}]}, \tag{8.15}$$

则根据数学期望的定义, 这等价于令

$$\begin{aligned} h_t(\boldsymbol{x}) &= \arg\max_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\frac{e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}]} f(\boldsymbol{x})h(\boldsymbol{x})\right] \\ &= \arg\max_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[f(\boldsymbol{x})h(\boldsymbol{x})\right] . \end{aligned} \tag{8.16}$$

由 $f(\boldsymbol{x}), h(\boldsymbol{x}) \in \{-1, +1\}$, 有

$$f(\boldsymbol{x})h(\boldsymbol{x}) = 1 - 2\,\mathbb{I}(f(\boldsymbol{x}) \neq h(\boldsymbol{x})) , \tag{8.17}$$

则理想的基学习器

$$h_t(\boldsymbol{x}) = \arg\min_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[\mathbb{I}(f(\boldsymbol{x}) \neq h(\boldsymbol{x}))\right] . \tag{8.18}$$

由此可见, 理想的 $h_t$ 将在分布 $\mathcal{D}_t$ 下最小化分类误差. 因此, 弱分类器将基于分布 $\mathcal{D}_t$ 来训练, 且针对 $\mathcal{D}_t$ 的分类误差应小于 0.5. 这在一定程度上类似“残差逼近”的思想. 考虑到 $\mathcal{D}_t$ 和 $\mathcal{D}_{t+1}$ 的关系, 有

$$\begin{aligned} \mathcal{D}_{t+1}(\boldsymbol{x}) &= \frac{\mathcal{D}(\boldsymbol{x})\,e^{-f(\boldsymbol{x})H_t(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_t(\boldsymbol{x})}\right]} \\ &= \frac{\mathcal{D}(\boldsymbol{x})\,e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}e^{-f(\boldsymbol{x})\alpha_t h_t(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_t(\boldsymbol{x})}\right]} \\ &= \mathcal{D}_t(\boldsymbol{x})\cdot e^{-f(\boldsymbol{x})\alpha_t h_t(\boldsymbol{x})}\frac{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\right]}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_t(\boldsymbol{x})}\right]} , \end{aligned} \tag{8.19}$$