中山大學
SUN YAT-SEN UNIVERSITY

第7章 集成学习

1. 个体与集成
2. 结合策略
3. Bagging与随机森林
4. Boosting

中山大學 SUN YAT-SEN UNIVERSITY

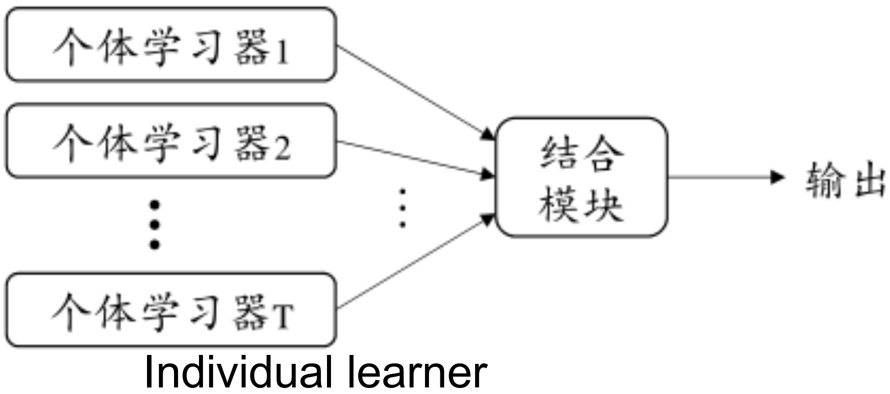
第7章 集成学习

1. 个体与集成
2. 结合策略
3. Bagging与随机森林
4. Boosting

概念——集成学习

- 集成学习（ensemble learning）

集成学习通过构建并结合多个学习器来完成学习任务，有时也被称为多分类器系统（multi-classifier system）、基于委员会的学习（committee-based learning）等。



- 同质（homogeneous）集成中个体学习器又称基学习器
- 异质（heterogeneous）集成中个体学习器又称组件学习器

（对数几率回归、感知机、神经网络、支持向量机、决策树）

个体与集成

简单举例

在二分类问题中，假定三个分类器在三个样本中的表现如下图所示，其中√表示分类正确，×号表示分类错误，集成的结果通过投票产生。

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

(a) 集群提升性能

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

(b) 集群不起作用

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

(c) 集群起负作用

集成个体应“好而不同”

个体与集成

简单分析

在二分类问题$y \in \{-1,1\}$和真实函数f中，假设通过简单投票法（若有超过半数的基分类器输出为正例，则集成的分类器就输出正例）来结合T个基分类器形成集成

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

假设基分类器的错误率为ϵ，即对每个基分类器h_i有

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

个体与集成

简单分析

假设基分类器的错误率相互独立，则由Hoeffding不等式可得集成的错误率为

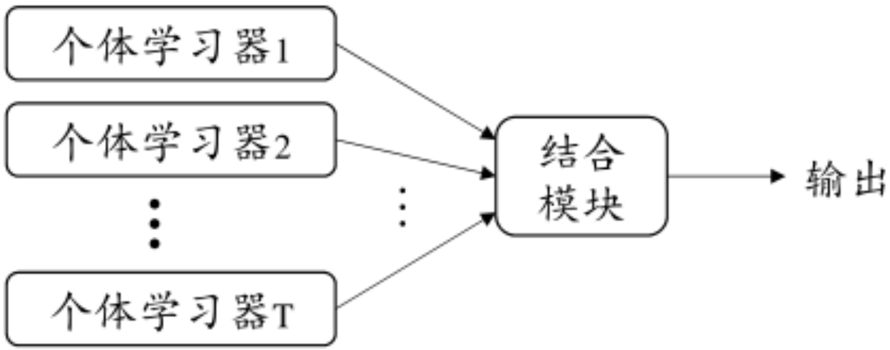
$$P(H(\boldsymbol{x}) \neq f(\boldsymbol{x})) = \sum_{k=0}^{\lfloor T/2 \rfloor} \binom{T}{k} (1-\epsilon)^k \epsilon^{T-k}$$

$$\leq \exp\left(-\frac{1}{2} T (1-2\epsilon)^2\right)$$

上式显示，在一定条件下，随着集成分类器数目的增加，集成的错误率将指数级下降，最终趋向于0

个体与集成

- 集成学习（ensemble learning）



Individual learner

准确性

&

多样性

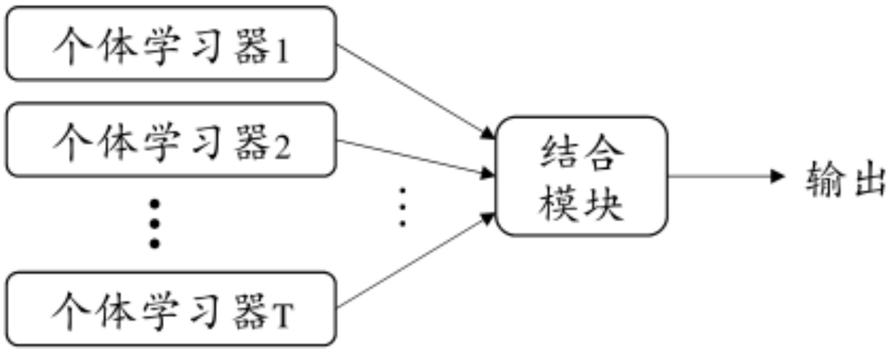
https://www.sohu.com/a/229614770_662288

集成学习的核心：如何产生并结合“好而不同”的个体学习器？

- 产生（怎样获得不同的弱分类器）
  - 序列化方法：个体学习器间存在强依赖关系、必须串行生成的方法，代表是Boosting（AdaBoost）
  - 并行化方法：个体学习器间不存在强依赖关系、可同时生成的方法，代表是Bagging和“随机森林”（random forest）

个体与集成

◻ 集成学习（ensemble learning）



Individual learner

准确性

&

多样性

https://www.sohu.com/a/229614770_662288

集成学习的核心：如何产生并结合“好而不同”的个体学习器？

◻ 结合（怎样组合弱分类器）

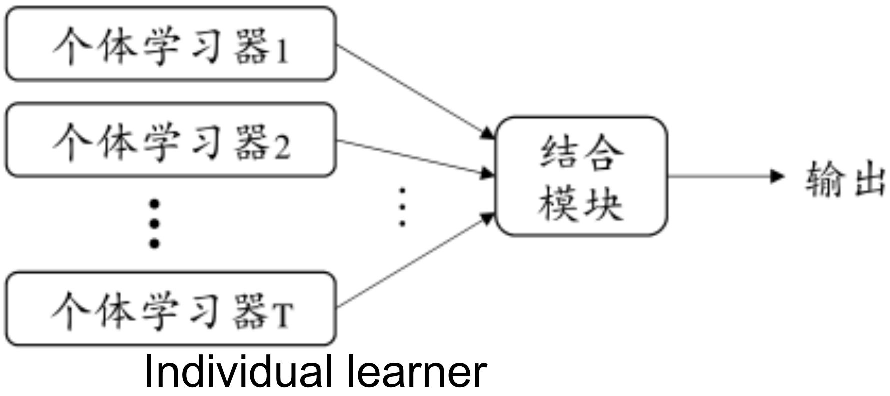
- 平均法（用于数值型输出，有简单平均法、加权平均法等）
- 投票法（用于分类任务，有绝对多数投票法、相对多数投票法、加权投票法等）
- 学习法（用于训练数据较多的情形，代表是Stacking）

第7章 集成学习

1. 个体与集成
2. 结合策略
3. Bagging与随机森林
4. Boosting

结合策略

- 集成学习（ensemble learning）



准确性
&
多样性

https://www.sohu.com/a/229614770_662288

集成学习的核心：如何产生并结合“好而不同”的个体学习器？

- 结合（怎样组合弱分类器）
  - 平均法（用于数值型输出，有简单平均法、加权平均法等）
  - 投票法（用于分类任务，有绝对多数投票法、相对多数投票法、加权投票法等）
  - 学习法（用于训练数据较多的情形，代表是Stacking）

平均法

□ 简单平均和加权平均

- 简单平均法

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

- 加权平均法

$$H(\boldsymbol{x}) = \frac{1}{T}\sum_{i=1}^{T} \omega_i h_i\,(\boldsymbol{x})\,, \omega_i \geq 0, \sum_{i=1}^{T} \omega_i = 1$$

平均法

投票法

- 绝对多数投票法（majority voting）

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

- 相对多数投票法（plurality voting）

$$H(\boldsymbol{x}) = c_{\arg\max_j \sum_{i=1}^{T} h_i^j(\boldsymbol{x})}$$

- 加权投票法（weighted voting）

$$H(\boldsymbol{x}) = c_{\arg\max_j \sum_{i=1}^{T} \omega_i h_i^j(\boldsymbol{x})}$$

$h_i^j(\boldsymbol{x}) \in \{0,1\}$，若$h_i$将样本预测为类别$j$则取值为1，其他为0

集成学习策略

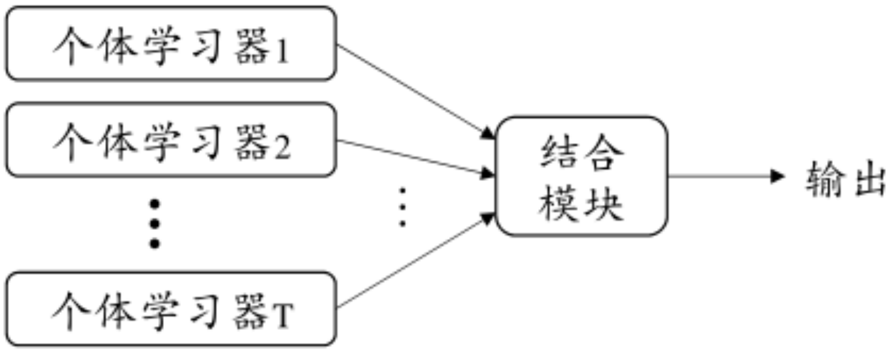
Bagging

Boosting

Bagging

Bagging

Bagging



Individual learner

准确性

&

多样性

https://www.sohu.com/a/229614770_662288

集成学习的核心：如何**产生**并**结合**“好而不同”的个体学习器？

产生（怎样获得不同的弱分类器）

- 序列化方法：个体学习器间存在强依赖关系、必须串行生成的方法，代表是Boosting（AdaBoost）
- **并行化方法：个体学习器间不存在强依赖关系、可同时生成的方法，代表是Bagging和“随机森林”（random forest）**

Bagging

- 训练数据集

$$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$$

- Bootstrap 自助法采样： D_{bs}
  - 独立随机放回

- Bagging算法流程
  - Step 1： 从大小为m的原始数据集D中随机抽取 1个数据并放回，重复m次，得到含有m个样本的采样集
  - Step 2: 重复上述操作T次，产生T个独立的数据集
  - Step 3: 利用上述各数据集训练出 T 个分类器
  - Step 4: 投票决定最终分类器

Bagging

Bagging

数据：$D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\cdots,(\boldsymbol{x}_m,y_m)\}$，$\boldsymbol{x}_i\in\mathbb{R}^d, y_i\in\mathbb{R}^?$

模型：$H(\boldsymbol{x})=\underset{y\in\mathcal{Y}}{\operatorname{argmax}}\sum_{t=1}^{T}\mathbb{I}(h_t(\boldsymbol{x})=y)$（分类任务）

$H(\boldsymbol{x})=\frac{1}{T}\sum_{t=1}^{T}h_t(\boldsymbol{x})$（回归任务）

输入: 训练集 $D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\ldots,(\boldsymbol{x}_m,y_m)\}$;
基学习算法 $\mathfrak{L}$;
训练轮数 T.

过程:

1: **for** $t=1,2,\ldots,T$ **do**
2: $h_t=\mathfrak{L}(D,\mathcal{D}_{bs})$
3: **end for**

输出: $H(\boldsymbol{x})=\underset{y\in\mathcal{Y}}{\arg\max}\sum_{t=1}^{T}\mathbb{I}(h_t(\boldsymbol{x})=y)$

Bagging算法描述

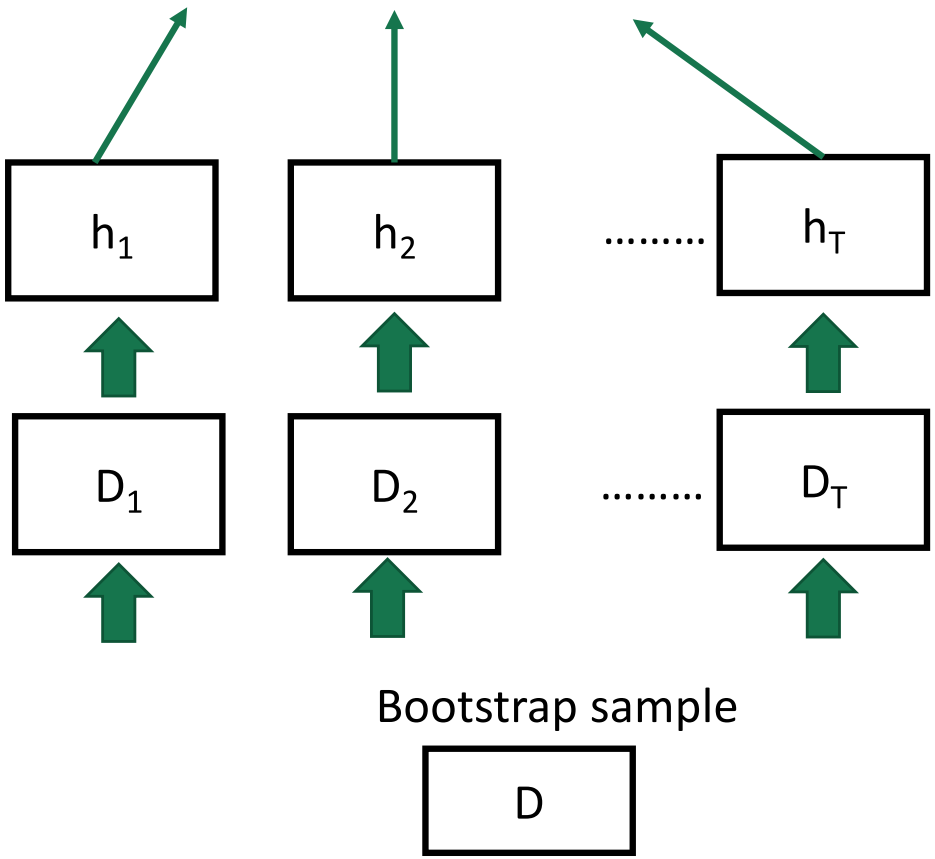
输入： 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
基学习算法 $\mathfrak{L}$;
训练轮数 T.

过程：

for $t = 1, 2, \ldots, T$ **do**
  $h_t = \mathfrak{L}(D, \mathcal{D}_{bs})$
end for

输出： $H(\boldsymbol{x}) = \underset{y \in \mathcal{Y}}{\arg\max} \sum_{t=1}^{T} \mathbb{I}(h_t(\boldsymbol{x}) = y)$

$$H(x) = argmax_{y \in Y} \sum_{t=1}^{T} \mathbb{I}(h_t(\boldsymbol{x}) = y)$$



Bagging算法特点

基本特点

- 个体学习器不存在强依赖关系
- 并行化生成
- 自助采样法

时间复杂度低

- 假定基学习器的计算复杂度为O(m)，采样与投票/平均过程的复杂度为O(s)，则bagging的复杂度大致为T(O(m)+O(s))
- 由于O(s)很小且T是一个不大的常数
- 因此训练一个bagging集成与直接使用基学习器的复杂度同阶

可使用包外估计 (Out-Of-Bag Estimate)

包外估计

- Bootstrapping
  - 原始数据集
  - Bootstrap 数据集

$$D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),..(\boldsymbol{x}_i,y_i),\dots,(\boldsymbol{x}_m,y_m)\}$$

$$D=\{(\boldsymbol{x}_1,y_1)',(\boldsymbol{x}_2,y_2)',..(\boldsymbol{x}_i,y_i)',\dots,(\boldsymbol{x}_m,y_m)'\}$$

对于任意个数据 i 的包外概率是？ $\left(1-\frac{1}{m}\right)^m$

$$\lim_{m\to\infty}\left(1-\frac{1}{m}\right)^m=\frac{1}{e}\approx 0.368$$

36.8%，
样本未在训练集中使用
可以作为测试集

	h_1	h_2	h_3	……	h_T
$(\boldsymbol{x}_1,y_1)$	o	X	o		o
$(\boldsymbol{x}_2,y_2)$	X	o	X		X
$(\boldsymbol{x}_3,y_3)$	X	X	X		o
……					
$(\boldsymbol{x}_m,y_m)$	o	X	X		o

包外估计

- $H^{oob}(x)$表示对样本x的包外预测，即仅考虑那些未使用样本x训练的基学习器在x上的预测

$$H^{oob}(\boldsymbol{x}) = \arg\max_{y \in \mathcal{Y}} \sum_{t=1}^{T} \mathbb{I}(h_t(\boldsymbol{x}) = y) \cdot \mathbb{I}(\boldsymbol{x} \notin D_t)$$

- Bagging泛化误差的包外估计为：

$$\epsilon^{oob} = \frac{1}{|D|} \sum_{(\boldsymbol{x},y) \in D} \mathbb{I}(H^{oob}(\boldsymbol{x}) \neq y)$$

Bagging

□ Bagging

数据：$D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\cdots,(\boldsymbol{x}_m,y_m)\}$，$\boldsymbol{x}_i\in\mathbb{R}^d, y_i\in\mathbb{R}^?$

模型：$H(\boldsymbol{x})=\underset{y\in\mathcal{Y}}{\operatorname{argmax}}\sum_{t=1}^{T}\mathbb{I}(h_t(\boldsymbol{x})=y)$（分类任务）

$H(\boldsymbol{x})=\frac{1}{T}\sum_{t=1}^{T}h_t(\boldsymbol{x})$（回归任务）

策略：个体学习器的产生采用并行化方法（采用自助采样法从包含m个样本的数据集中，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），个体学习器的结合采用简单投票法（分类任务）或简单平均法（回归任务）

算法：各个个体学习器所对应的算法

Bagging

- Bagging

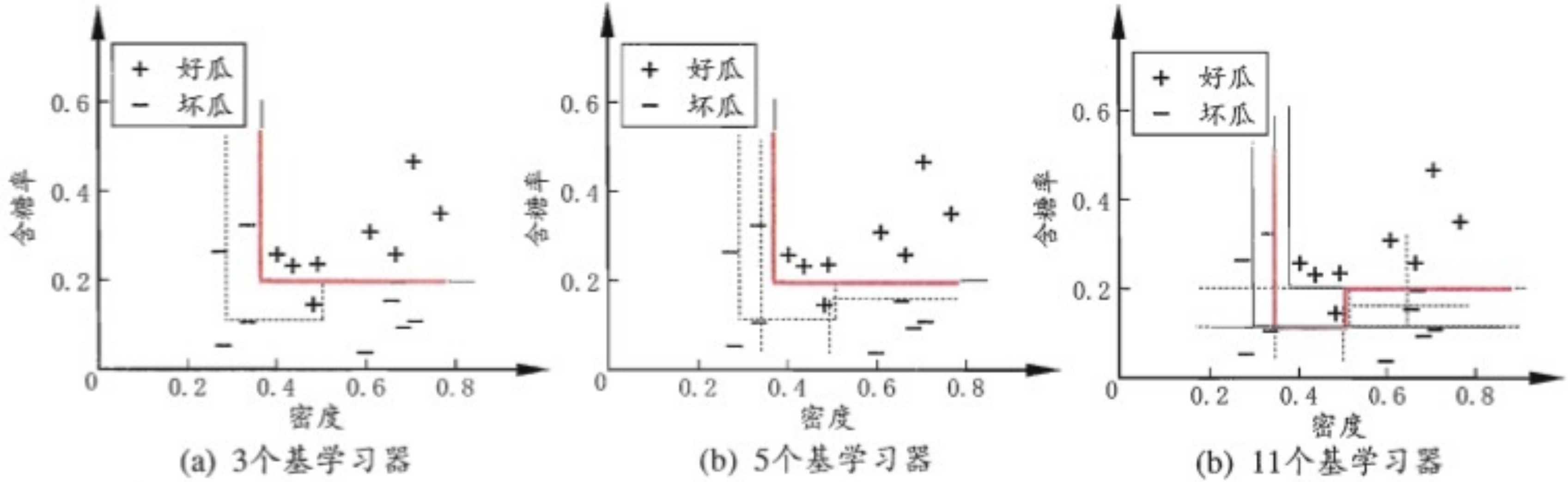


图 8.6 西瓜数据集 3.0α 上 Bagging 集成规模为 3、5、11 时，集成(红色)与基学习器(黑色)的分类边界.

从偏差-方差的角度：降低方差，在不剪枝的决策树、神经网络等易受样本影响的学习器上效果更好

随机森林

□ 随机森林

数据： $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$，$\boldsymbol{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}^?$

模型： $H(\boldsymbol{x}) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \sum_{t=1}^{T} \mathbb{I}(h_t(\boldsymbol{x}) = y)$（分类任务）

$H(\boldsymbol{x}) = \frac{1}{T} \sum_{t=1}^{T} h_t(\boldsymbol{x})$（回归任务）

策略： 个体学习器的产生采用并行化方法（采用自助采样法从包含m个样本的数据集中，采样出T个含m个训练样本的采样集；然后以决策树为基学习器，在基决策树的训练过程中引入了随机属性选择；对于每个采样集训练出一个基学习器），个体学习器的结合采用简单投票法（分类任务）或简单平均法（回归任务）

算法： 各个个体学习器所对应的算法

Bagging与随机森林

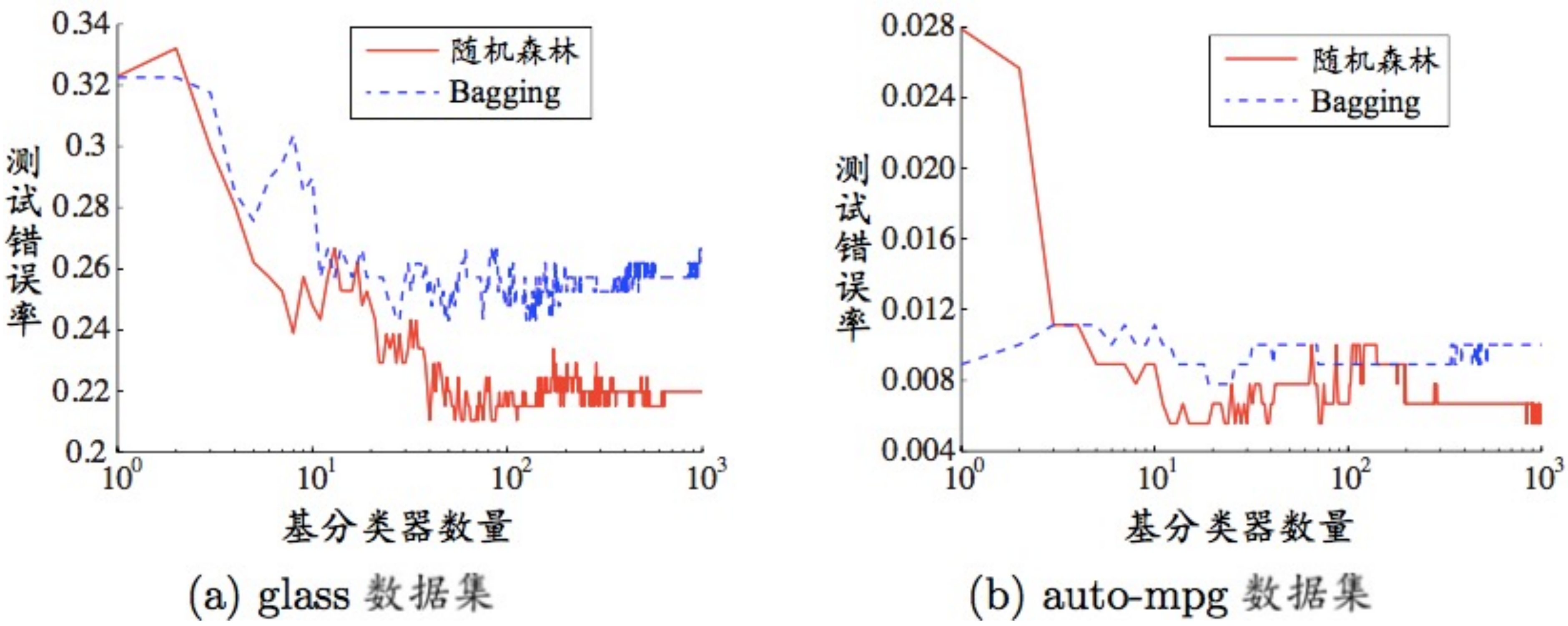
随机森林

- ✓ 随机森林是bagging的一个扩展变种
- ✓ 采样的随机性（自助采样法）
- ✓ 属性选择的随机性（基决策树最优属性的选择）

- ➢ 对基决策树的每个结点，先从该结点的属性集合中随机选择一个包含k个属性的子集
- ➢ 然后再从这个自己中选择一个最优属性用于划分

这里的参数k控制随机性的引入程度：若令$k = d$，则该决策树的构建与传统决策树相同；若令$k = 1$，则是随机选择一个属性用于划分；一般情况下，推荐值$k = \log_2 d$

Bagging与随机森林



- 随机森林的收敛性与Bagging相似
- 随机森林简单、容易实现、计算开销小，令人惊奇的是，它在很多现实任务中展现出强大的性能，被誉为“代表集成学习技术水平的方法”

Boosting

Boosting

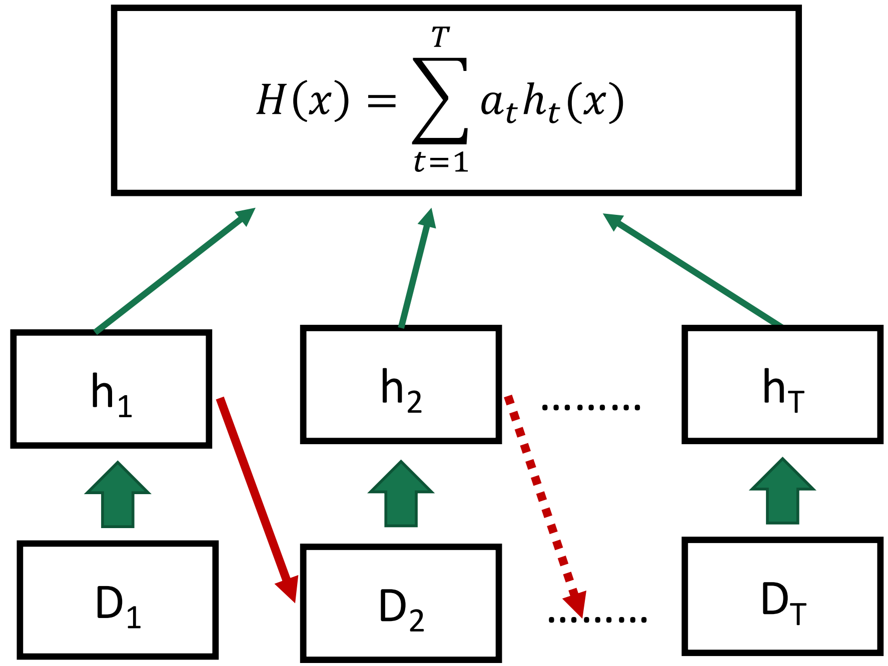
- Boosting（提升）方法

  Boosting是一种可将弱学习器提升为强学习器的算法。如在分类问题中，它通过改变训练样本的权重，学习多个分类器，并将这些分类器进行线性组合，提高分类的性能。

- 工作机制
  - 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关注
  - 从调整样本分布后的训练集训练出下一个基学习器
  - 重复进行，直至基学习器数目达到事先指定的值T
  - 将这T个基学习器进行加权结合

Boosting

- 基本框架
  - 产生第一个 分类器 $h_1(\boldsymbol{x})$
  - 产生第二个分类器 $h_2(\boldsymbol{x})$ 来补足 $h_1(\boldsymbol{x})$
  - 找到第三个
  - ……….
  - 整合

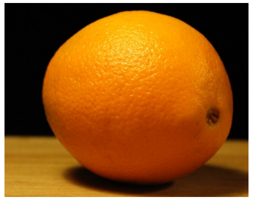


- 顺序过程，每一个后续模型都会尝试纠正先前模型的错误。后续模型依赖之前的模型。串行生成
- 个体学习器存在强依赖关系
- 每次调整训练数据的样本分布

调整训练数据的样本分布

训练一个分类器判别苹果

$(\boldsymbol{x}_1, y_1)$



$(\boldsymbol{x}_2, y_2)$

$(\boldsymbol{x}_3, y_3)$

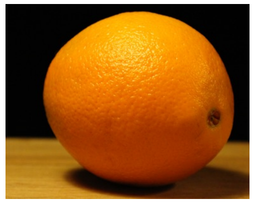
$(\boldsymbol{x}_4, y_4)$

$(\boldsymbol{x}_5, y_5)$

$(\boldsymbol{x}_6, y_6)$

$(\boldsymbol{x}_7, y_7)$

$(\boldsymbol{x}_8, y_8)$



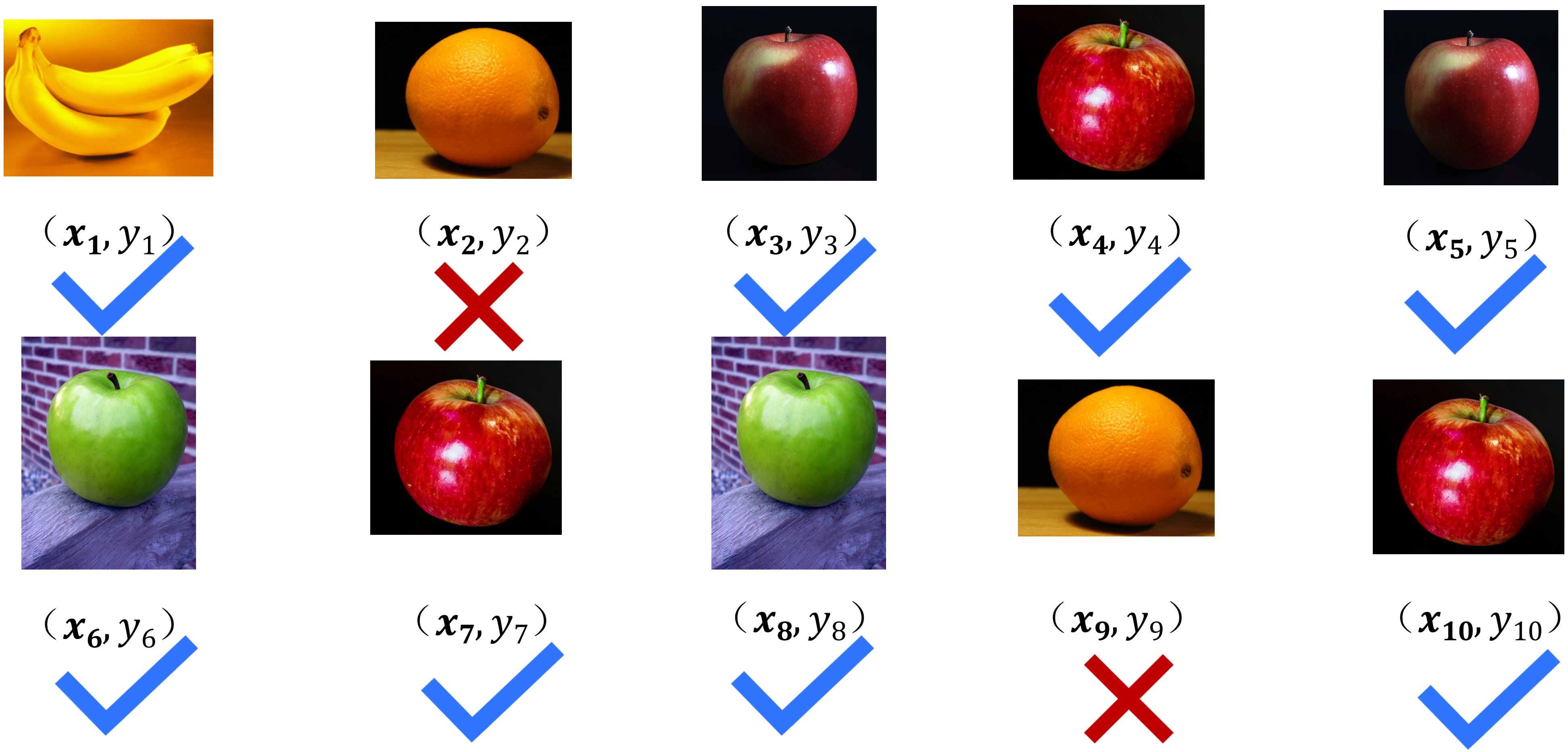
$(\boldsymbol{x}_9, y_9)$

$(\boldsymbol{x}_{10}, y_{10})$

苹果是圆形的

调整训练数据的样本分布

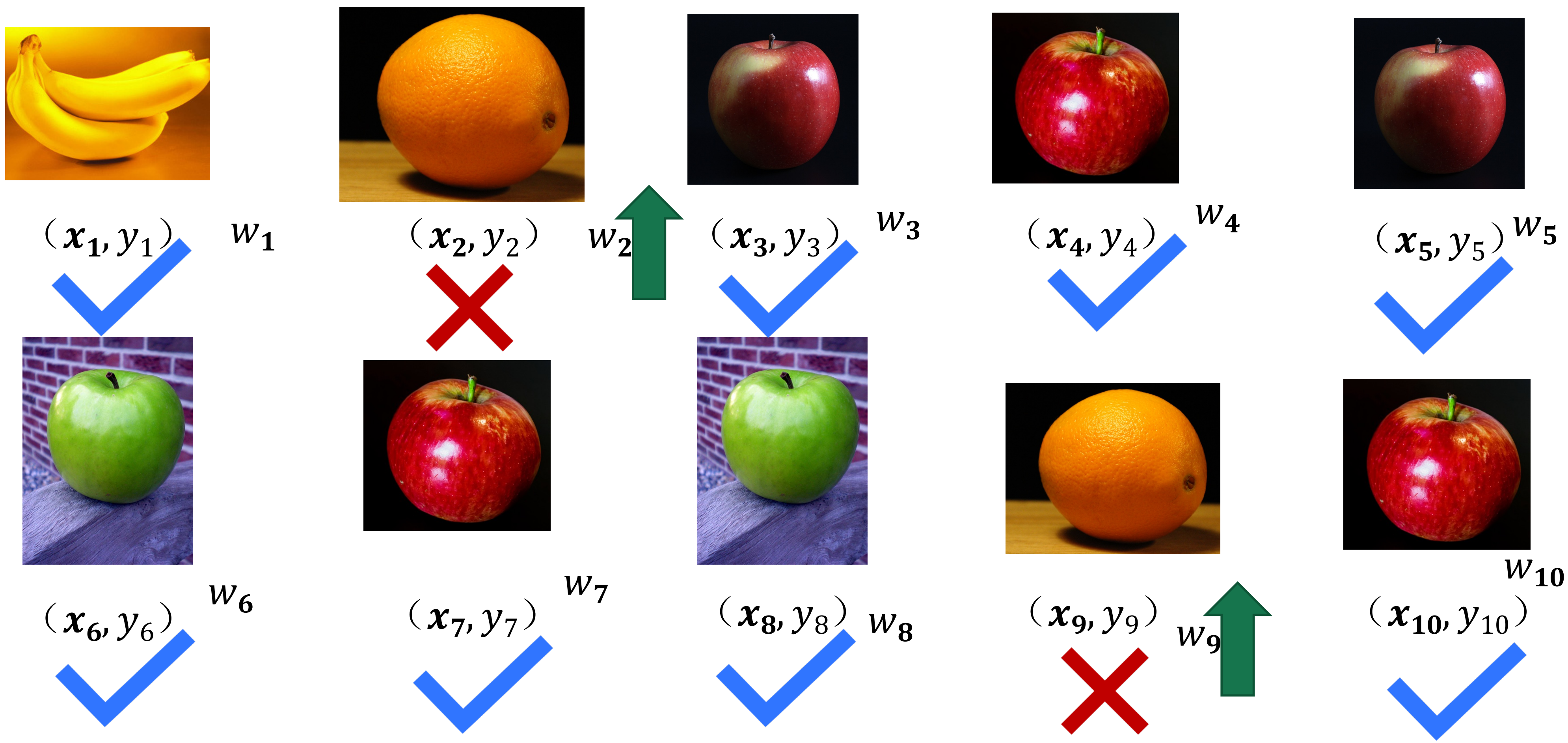
- 训练一个分类器判别苹果



苹果是圆形的 Boosting会怎么做？

调整训练数据的样本分布

- 训练一个分类器判别苹果



加权

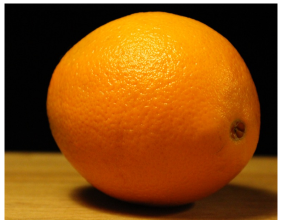
调整训练数据的样本分布

- Reweighting （重赋权值）

（$\boldsymbol{x}_1, y_1, w_1$） w_1=1 w_1=0.5

（$\boldsymbol{x}_2, y_2, w_2$） w_2=1 w_2=0.5

（$\boldsymbol{x}_3, y_3, w_3$） w_3=1 w_3=0.5



（$\boldsymbol{x}_4, y_4, w_4$） w_4=1 w_4=1.5

……………

Boosting - Boosting算法

Input: Sample distribution $\mathcal{D}$;
Base learning algorithm $\mathfrak{L}$;
Number of learning rounds T.

Process:

1. $\mathcal{D}_1 = \mathcal{D}$. % Initialize distribution
2. **for** $t = 1, \ldots, T$:
3. $h_t = \mathfrak{L}(\mathcal{D}_t)$; % Train a weak learner from distribution $\mathcal{D}_t$
4. $\epsilon_t = P_{\boldsymbol{x} \sim D_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$; % Evaluate the error of h_t
5. $\mathcal{D}_{t+1} = Adjust\_Distribution(\mathcal{D}_t, \epsilon_t)$
6. **end**

Output: $H(\boldsymbol{x}) = Combine\_Outputs(\{h_1(\boldsymbol{x}), \ldots, h_t(\boldsymbol{x})\})$

- Boosting族算法最著名的代表是AdaBoost

Boosting

- AdaBoost

$$\underbrace{\underbrace{\textbf{Ada}}_{\text{Adaptive}}\ \underbrace{\textbf{Boost}}_{\text{Boosting}}}_{\text{A learning algorithm}}$$

Building a strong classifier from a lot of weaker ones

AdaBoost算法基本思想

基学习器的线性组合

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

损失函数

$$\ell(H|D) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}[H(x_i) \neq y_i]$$

$$H(x_i) \neq y_i \quad \Rightarrow \quad y_i H(\boldsymbol{x_i}) \leq 0 \quad \Rightarrow \quad e^{-y_i H(\boldsymbol{x_i})} \geq 1$$

$$\mathbb{I}[H(x_i) \neq y_i] \leq e^{-y_i H(\boldsymbol{x_i})}$$

最小化指数损失函数

$$\ell(H|D) = \mathbb{E}_{\boldsymbol{x}\sim D}\mathbb{I}[H(x_i) \neq y_i] \;\; \leq \mathbb{E}_{\boldsymbol{x}\sim D}(e^{-y_i H(\boldsymbol{x_i})})$$

$$\ell_{exp}(H|D) = \mathbb{E}_{\boldsymbol{x}\sim D}(e^{-y_i H(\boldsymbol{x_i})})$$

AdaBoost算法基本思想

$$\ell_{exp}(H|D) = \mathbb{E}_{\boldsymbol{x}\sim D}(e^{-y_i H(\boldsymbol{x}_i)})$$

- 当基分类器h_t基于分布D_t产生后，该基分类器的权重α_t应使得$\alpha_t h_t$最小化指数损失函数

$$\begin{aligned}
\ell_{exp}(a_t h_t|D_t) &= \mathbb{E}_{\boldsymbol{x}\sim D^t}(e^{-y_i a_t h_t}) \\
&= \mathbb{E}_{\boldsymbol{x}\sim D^t}\{\ e^{-a_t}\mathbb{I}[h_t(x_i) = y_i] + e^{a_t}\mathbb{I}[h_t(x_i) \neq y_i]\ \} \\
&= e^{-a_t}\mathrm{P}_{\boldsymbol{x}\sim D^t}[h_t(x_i) = y_i] + e^{a_t}\mathrm{P}_{\boldsymbol{x}\sim D^t}[h_t(x_i) \neq y_i] \\
&= e^{-a_t}(1-\epsilon_t) + e^{a_t}\epsilon_t
\end{aligned}$$

$$\epsilon_t = \mathrm{P}_{\boldsymbol{x}\sim D^t}[h_t(x_i) \neq y_i]$$

- 令指数损失函数的导数为0，即

$$\frac{\vartheta \ell_{exp}(a_t h_t|D_t)}{\vartheta a_t} = -e^{-a_t}(1-\epsilon_t) + e^{a_t}\epsilon_t \qquad a_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$$

AdaBoost算法基本思想

- 基本想法：通过调整权值，产生一个新的样本分布及h

$$a^t = \frac{1}{2}\ln((1-\varepsilon_t)/\varepsilon_t)$$

- IF 数据 i 在 $h_t(\boldsymbol{x})$ 判断失败

增大 $w_i{}^t$ to $w_i{}^{t+1}$

$$w_i{}^{t+1} = w_i{}^t * \exp(a_t)$$

- IF 数据 i 在 $h_t(\boldsymbol{x})$ 判断成功

降低 $w_i{}^t$ to $w_i{}^{t+1}$

$$w_i{}^{t+1} = w_i{}^t * \exp(-a_t)$$

$$w_i{}^{t+1} = w_i{}^t * \exp(-\boldsymbol{f}(\boldsymbol{x})\boldsymbol{h_t}(\boldsymbol{x})a_t)$$

Boosting

AdaBoost

输入： $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$， $\boldsymbol{x}_i \in \mathbb{R}^d, y_i \in \{-1,1\}$

输出： $F(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right)$

步骤1：初始化训练数据的起始权值分布

$$\mathcal{D}_1(\boldsymbol{x}_1) = \mathcal{D}_1(\boldsymbol{x}_2) = \cdots = \mathcal{D}_1(\boldsymbol{x}_m) = \frac{1}{m}$$

步骤2：对T个个体学习器，依次序列化进行以下步骤

$$\boxed{\text{2:} \ \ \textbf{for } t = 1, 2, \ldots, T \ \ \textbf{do}}$$

AdaBoost

步骤2.1：使用具有样本分布$\mathcal{D}_t$的训练数据D，训练得到个体分类器

$$h_t(\boldsymbol{x}): \mathcal{X} \rightarrow \mathcal{Y} = \{-1,1\}$$

步骤2.2：计算$h_t(\boldsymbol{x})$的训练误差

$$\epsilon_t = P_{\boldsymbol{x}\sim\mathcal{D}_t}(h_t(\boldsymbol{x}) \neq y) = \sum_{i=1}^{m} \mathcal{D}_t(\boldsymbol{x}_i)\mathbb{I}(h_t(\boldsymbol{x}_i) \neq y_i)$$

☐ AdaBoost

步骤2.3：计算$h_t(\boldsymbol{x})$在集成中的权重α_t

$$\alpha_t = \frac{1}{2}\ln\frac{1-\epsilon_t}{\epsilon_t}$$

步骤2.4：更新训练数据集的样本分布$\mathcal{D}_{t+1}(\boldsymbol{x}_i), i = 1, \cdots, m$

$$\mathcal{D}_{t+1}(\boldsymbol{x}_i) = \frac{\mathcal{D}_t(\boldsymbol{x}_i)}{Z_t} \times \begin{cases} \exp(-\alpha_t), if\ h_t(\boldsymbol{x}_i) = y_i \\ \exp(\alpha_t), if\ h_t(\boldsymbol{x}_i) \neq y_i \end{cases}$$

$$= \frac{\mathcal{D}_t(\boldsymbol{x}_i)\exp(-\alpha_t h_t(\boldsymbol{x}_i) y_i)}{\sum_{i=1}^m \mathcal{D}_t(\boldsymbol{x}_i)\exp(-\alpha_t h_t(\boldsymbol{x}_i) y_i)}$$

$$Z_t = \sum_{i=1}^m {w_i}^t$$

Boosting

□ AdaBoost

步骤3：构建弱分类器的线性组合并得到最终的集成分类器

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

注意事项：

- 对于无法接受带权样本的个体学习器，可通过“重采样法”（re-sampling）来处理
- 对于误差$\epsilon_t > 0.5$的个体学习器，需要抛弃该个体学习器，采用“重采样法”来重启动，避免训练过程过早停止

Boosting – AdaBoost算法

输入: 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
基学习算法 $\mathfrak{L}$;
训练轮数 T.

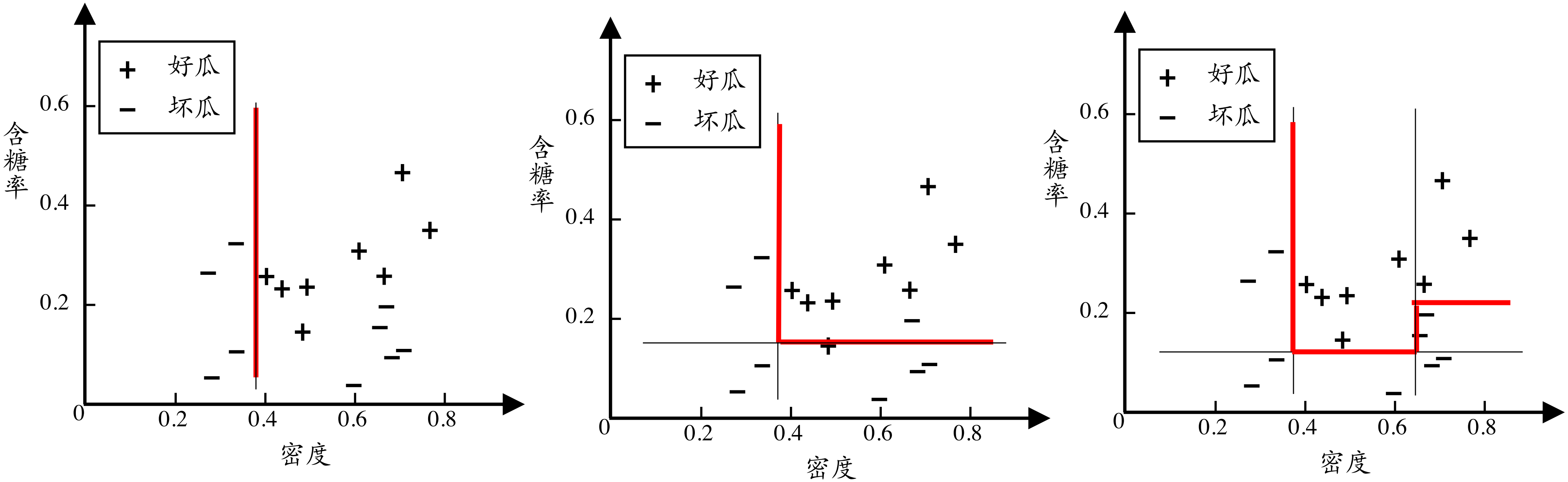
过程:

1: $\mathcal{D}_1(\boldsymbol{x}) = 1/m$.
2: **for** $t = 1, 2, \ldots, T$ **do**
3: $\quad h_t = \mathfrak{L}(D, \mathcal{D}_t)$;
4: $\quad \epsilon_t = P_{\boldsymbol{x} \sim \mathcal{D}_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$;
5: $\quad$ **if** $\epsilon_t > 0.5$ **then break**
6: $\quad \alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$;
7: $\quad$ $$\begin{aligned} \mathcal{D}_{t+1}(\boldsymbol{x}) &= \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t} \times \begin{cases} \exp(-\alpha_t), & \text{if } h_t(\boldsymbol{x}) = f(\boldsymbol{x}) \\ \exp(\alpha_t), & \text{if } h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}) \end{cases} \\ &= \frac{\mathcal{D}_t(\boldsymbol{x})\exp(-\alpha_t f(\boldsymbol{x}) h_t(\boldsymbol{x}))}{Z_t} \end{aligned}$$
8: **end for**

输出: $H(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right)$

$$Z_t = \sum_{i=1}^{m} w_i{}^t$$

AdaBoost例子



(a) t=1　(b) t=2　(c) t=3

- 从偏差-方差的角度：降低偏差，可对泛化性能相当弱的学习器构造出很强的集成

谢谢！