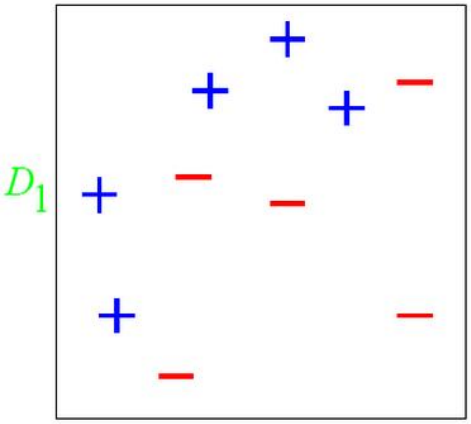
厦门大学
XIAMEN UNIVERSITY

集成学习

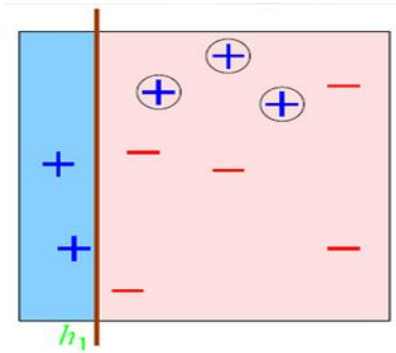
谭　忠

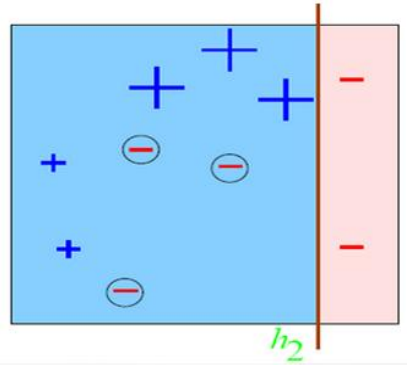
数学科学学院
SCHOOL OF MATHEMATICAL SCIENCES

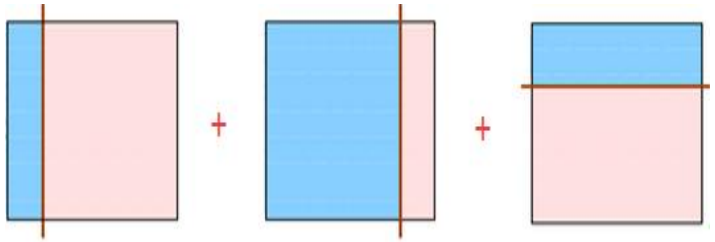
简单直观的例子

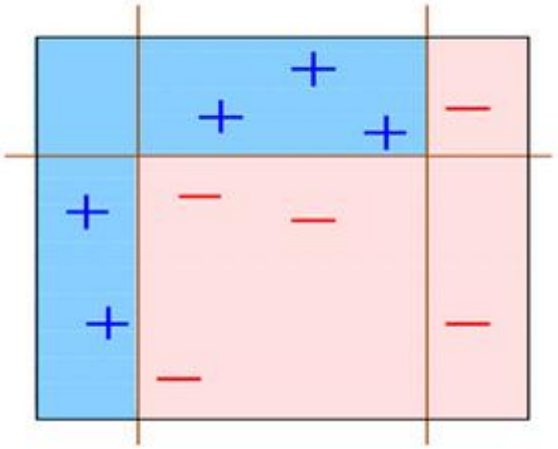


对实例进行分类









对多个分类器的分类结果进行某种**组合**来决定最终的分类，以取得比单个分类器更好的性能

集成学习

- 出于多种原因，机器学习模型可能会彼此不同。不同的机器学习模型可以对总体数据的不同样本进行操作，可以使用不同的建模技术，并且使用不同的假设。

集成学习

- 如果你加入由不同专业人员组成的团队，假设你正在和其他成员一起讨论一个技术主题。大家都只对自己的专业有所了解，而对其他专业技术一无所知。但是，如果最终能将这些技术知识组合在一起，将会对更多领域有更准确的猜测，这是集成学习的原理，也就是**结合不同个体模型(团队成员)的预测以提高准确性，并最大程度地减少错误。**

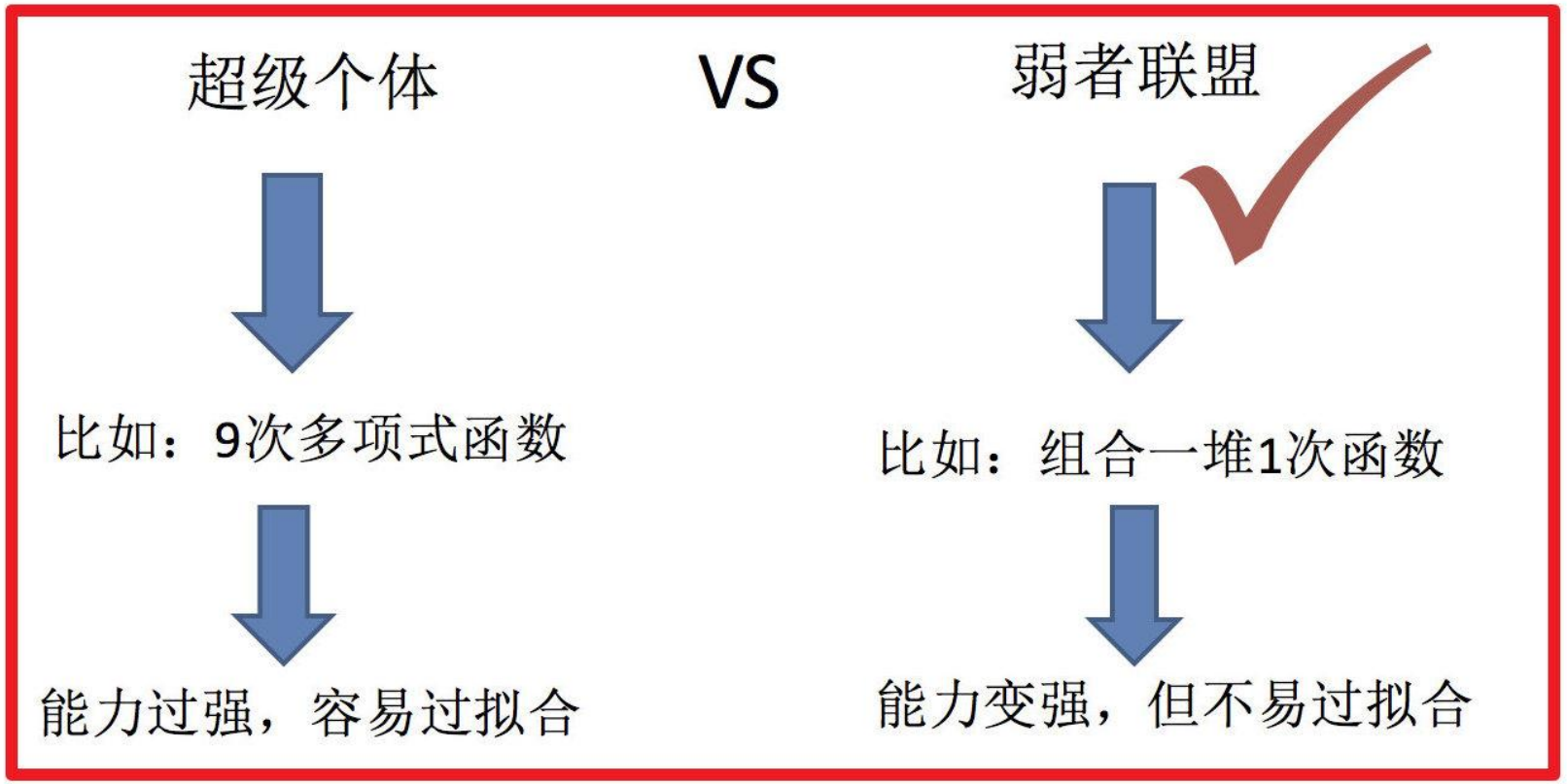
集成学习

- 统计学家已经证明，当一群人被要求用一系列可能的答案来猜测一个给定问题的正确答案时，他们所有的答案都会形成一个概率分布。真正知道正确答案的人会自信地选择正确的答案，而选择错误答案的人会将他们的猜测分散到可能的错误答案范围内。

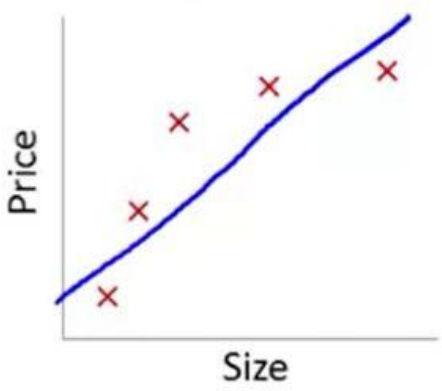
集成学习

- 所有的模型都有一定的误差。一个模型的误差将不同于另一个模型产生的误差，因为模型本身由于上述原因而不同。当检查所有的错误时，它们不会聚集在某一个答案周围，而是广泛分布。不正确的猜测基本上分散在所有可能的错误答案上，并**相互抵消**。与此同时，来自不同模型的正确猜测将聚集在正确的答案周围。当使用**集成训练方法**时，可以找到更可靠的正确答案。

什么是集成学习

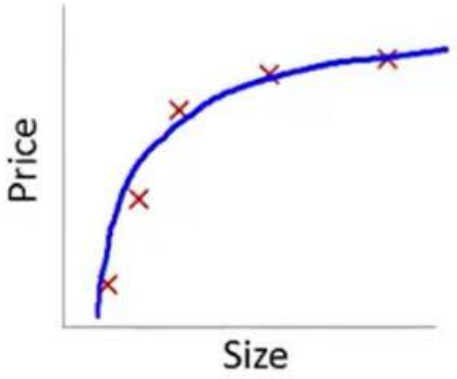


什么是集成学习



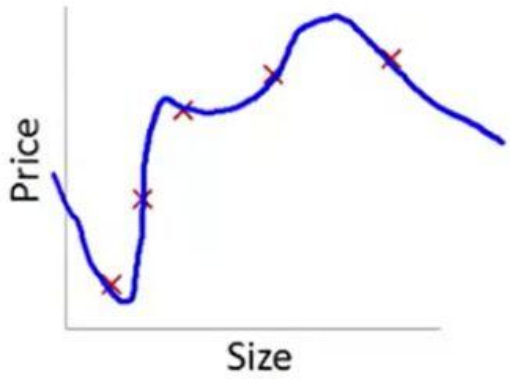
$\theta_0 + \theta_1 x$

欠拟合



$\theta_0 + \theta_1 x + \theta_2 x^2$

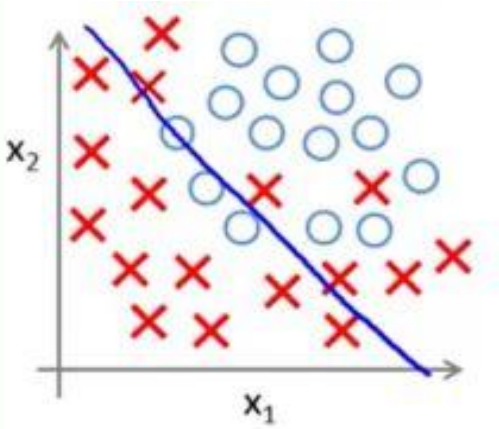
拟合



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

过拟合

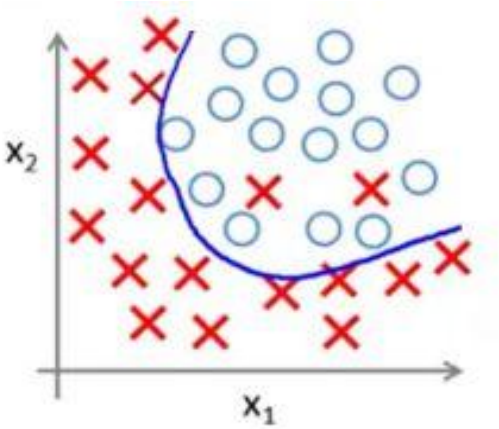
什么是集成学习



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

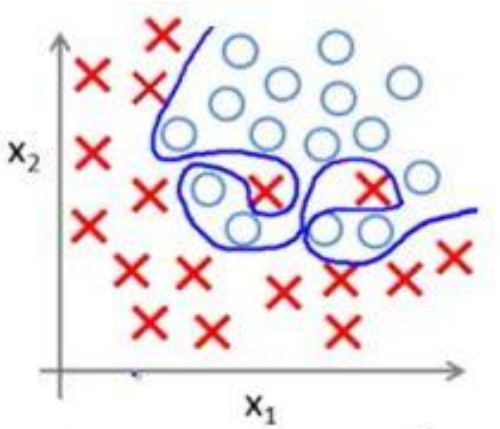
(g = sigmoid function)

欠拟合



$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$$

拟合



$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$$

过拟合

什么是集成学习

- 集成学习通过建立几个模型来解决单一预测问题。它的工作原理是生成多个分类器/模型，各自独立地学习和作出预测。这些预测最后结合成组合预测，因此优于任何一个单分类的做出预测。

集成学习

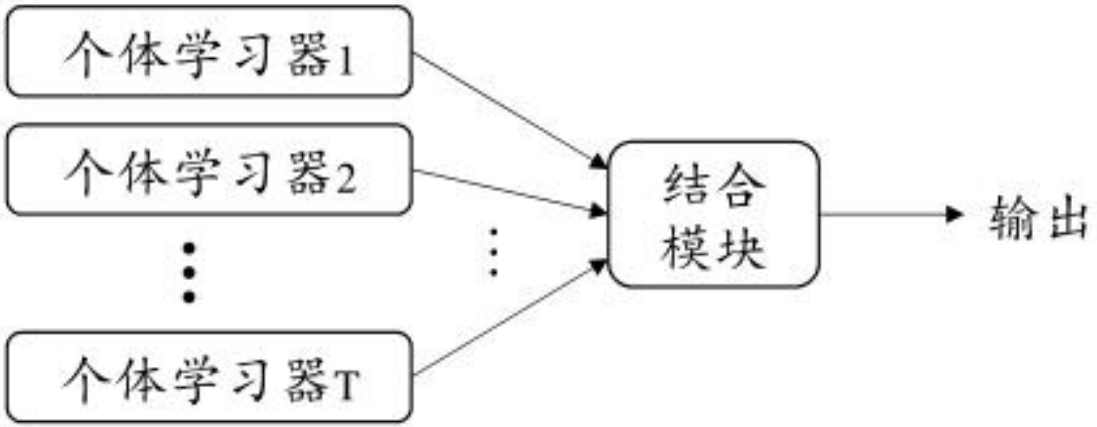
- 1、个体与集成
- 2、Boosting
  - Adaboost
- 3、Bagging与随机森林
- 4、结合策略
  - 平均法
  - 投票法
  - 学习法
- 5、多样性
  - 误差-分歧分解
  - 多样性度量
  - 多样性增强

Part 1

个体与集成

个体与集成

- **集成学习**(ensemble learning)通过构建并结合多个学习器来提升性能
- 集成学习的一般结构：先产生一组“个体学习器”，再用某种策略将它们结合起来



集成学习示意图

个体与集成

- 考虑一个简单的例子，在二分类问题中，假定3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

个体与集成

- 在图(a) 中, 每个分类器都只有66.6%的精度, 但集成学习却达到了100%; 在图(b)中, 三个分类器没有差别, 集成之后性能没有提高; 在图(c)中, 每个分类器的精度都只有33.3%, 集成学习的结果变得更糟.
- 这个简单的例子显示出：要获得好的集成, 个体学习器应 **“好而不同”**, 即个体学习器要有一定的“准确性”, 即学习器不能太坏, 并且要有“多样性”, 即学习器间具有差异.

个体与集成 – 简单分析

- 考虑二分类问题 $y \in \{-1, +1\}$ 和真实函数 f, 假定基分类器的错误率为 ϵ, 即对每个基分类器 h_i 有:

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

- 假设集成通过简单投票法结合 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:

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

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：

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

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

上式显示出，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.

个体与集成 – 简单分析

- 上面的分析有一个关键假设：**基学习器的误差相互独立**。现实任务中，个体学习器是为解决同一个问题训练出来的，显然不可能互相独立。事实上，个体学习器的“准确性”和“多样性”本身就存在冲突。
- **如何产生“好而不同”的个体学习器**是集成学习研究的核心

个体与集成 – 简单分析

- 根据个体学习器的生成方式, 目前的集成学习方法大致可分为两大类, 即个体学习器间存在强依赖关系、必须串行生成的序列化方法, 以及个体学习器间不存在强依赖关系、可同时生成的并行化方法;
- 前者的代表是 **Boosting**，后者的代表是 **Bagging** 和“**随机森林**” (Random Forest).

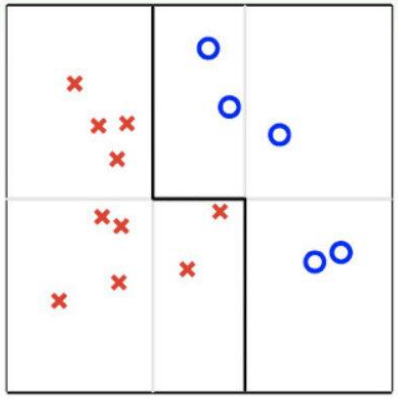
集成学习

- 机器学习的两个核心任务
- 任务一：如何优化训练数据 —> 主要用于解决**欠拟合**问题
- 任务二：如何提升泛化性能 —> 主要用于解决**过拟合**问题

集成学习中boosting和Bagging

- 只要单分类器的表现不太差，集成学习的结果总是要好于单分类器的

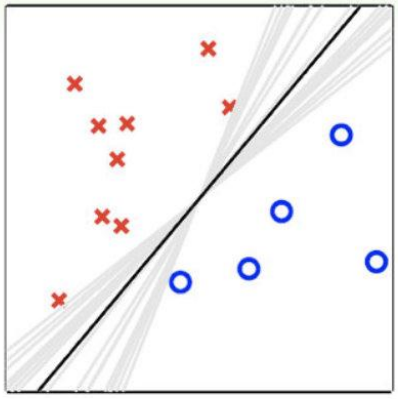
欠拟合问题



弱弱组合变强

主要方法：boosting逐步增强学习

过拟合问题



互相扼制变壮

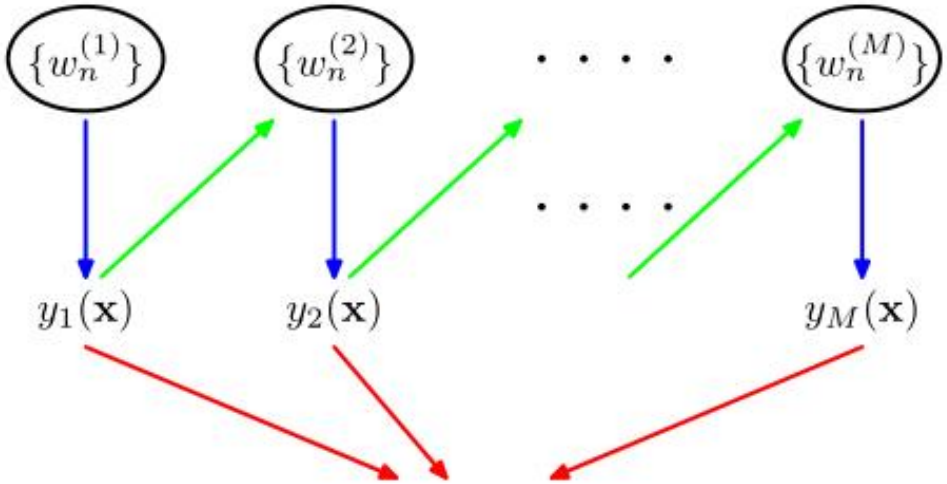
主要方法：Bagging采样学习集成

Part 2

Boosting

Boosting

- 个体学习器存在强依赖关系
- 串行生成
- 每次调整训练数据的样本分布



$$Y_M(\mathbf{x}) = \text{sign}\left(\sum_m^M \alpha_m y_m(\mathbf{x})\right)$$

Boosting

- Boosting是一族可将**弱学习器**提升为**强学习器**的算法.
- 这族算法的工作机制类似: 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，最终将这个基学习器进行加权结合.

Boosting - AdaBoost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: $\quad \mathcal{D}_{t+1}(\boldsymbol{x}) = \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t} \times \begin{cases} \exp(-\alpha_t), & \text{if } h_t(\boldsymbol{x}) = f(\boldsymbol{x}) \\ \exp(\alpha_t), & \text{if } h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}) \end{cases}$
$= \frac{\mathcal{D}_t(\boldsymbol{x}) \exp(-\alpha_t f(\boldsymbol{x}) h_t(\boldsymbol{x}))}{Z_t}$
8: **end for**

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

Boosting – AdaBoost推导

- 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

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

Boosting – AdaBoost推导

若H(x)能令指数损失函数最小化，则上式对H(x)的偏导值为0，即

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

令上式为0，可解得

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

Boosting – AdaBoost推导

因此，有

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

Boosting – AdaBoost推导

这意味着 sign(H(x)) 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.

Boosting – AdaBoost推导

- 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

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

其中

$$
\epsilon_t = P_{\boldsymbol{x}\sim\mathcal{D}_t}\left(h_t(\boldsymbol{x})\neq f(\boldsymbol{x})\right)
$$

Boosting – AdaBoost推导

- 指数损失函数的导数为

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

令其为0，解得

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

Boosting – AdaBoost推导

- 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

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

- 泰勒展开近似为

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

Boosting – AdaBoost推导

- 于是，理想的基学习器：

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

- 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：

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

Boosting – AdaBoost推导

- 根据数学期望的定义，这等价于令:

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

$$= \arg\max_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t} [f(\boldsymbol{x})h(\boldsymbol{x})] \ .$$

- 由 f(x),h(x)∈{-1,+1}, 有

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

Boosting – AdaBoost推导

- 则理想的基学习器

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

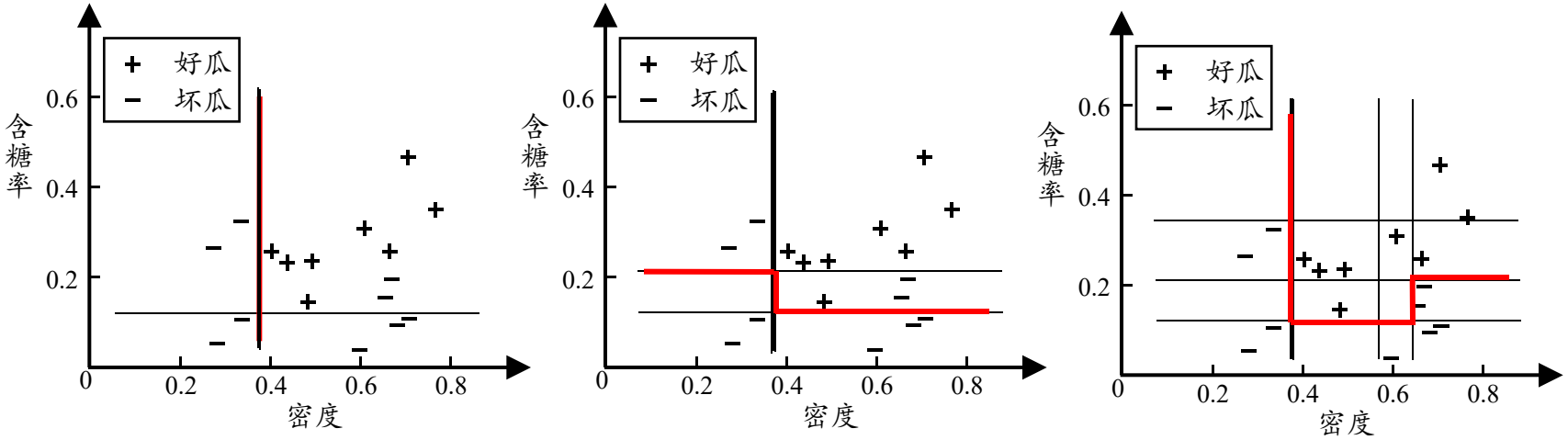
- 最终的样本分布更新公式

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

Boosting – AdaBoost 注意事项

- 数据分布的学习
  - 重赋权法：即在训练过程的每一轮中, 根据样本分布为每个训练样本重新赋予一个权重. 对无法接受带权样本的基学习算法
  - 重采样法：在每一轮学习中, 根据样本分布对训练集重新进行采样, 再用重采样而得的样本集对基学习器进行训练
- 重启动，避免训练过程过早停止

Boosting – AdaBoost实验



(a) 3个基学习器　(b) 5个基学习器　(c) 11个基学习器

西瓜数据集 3.0α 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.

- 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

Par3

Bagging与随机森林

Bagging与随机森林

- 个体学习器不存在强依赖关系
- 并行化生成
- 自助采样法

Bagging算法

Bagging [Breiman, 1996a] 是并行式集成学习方法最著名的代表.

给定包含 m 个样本的数据集, 我们先随机取出一个样本放入采样集中, 再把该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, 这样, 经过 m 次随机采样操作, 我们得到含 m 个样本的采样集, 初始训练集中有的样本在采样集里多次出现, 有的则从未出现. 初始训练集中约有63.2%的样本出现在采样集中.

Bagging算法

照这样, 我们可采样出 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. 这就是 Bagging 的基本流程.

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. 若分类预测时出现两个类收到同样票数的情形, 则最简单的做法是随机选择一个，也可进一步考察学习器投票的置信度来确定最终胜者.

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

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

Bagging算法特点

- 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

包外估计

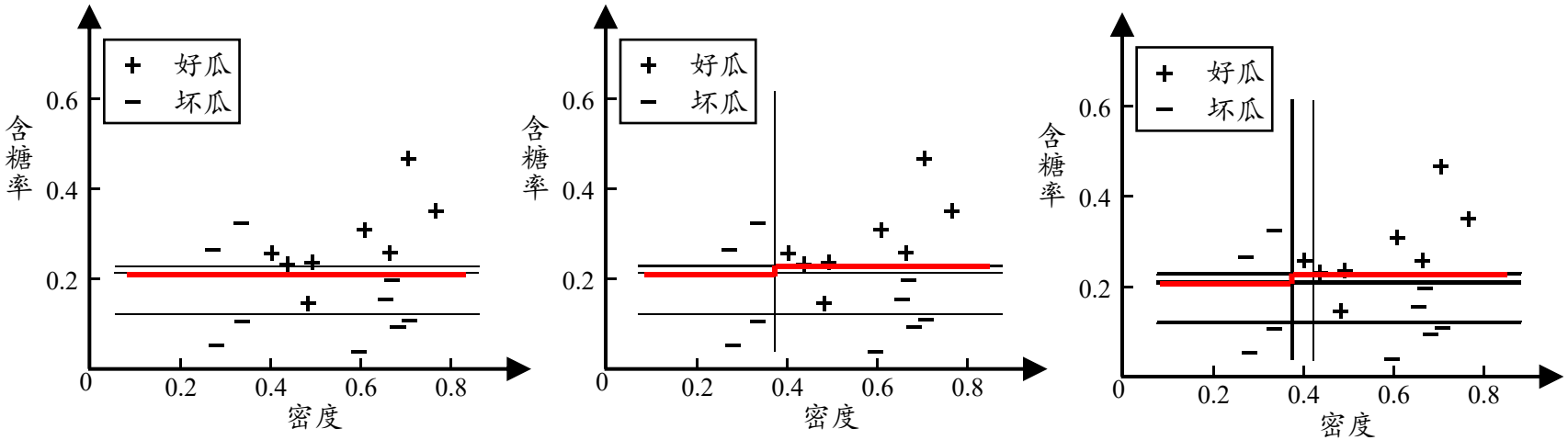
- H^{oob} (x) 表示对样本x的包外预测, 即仅考虑那些未使用x训练的基学习器在x上的预测,

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

- 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实验



(a) 3个基学习器　(b) 5个基学习器　(c) 11个基学习器

- 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(Random Forest，简称RF)是bagging的一个扩展变种。RF 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**.
- 采样的随机性
- 属性选择的随机性

随机森林算法

随机森林算法

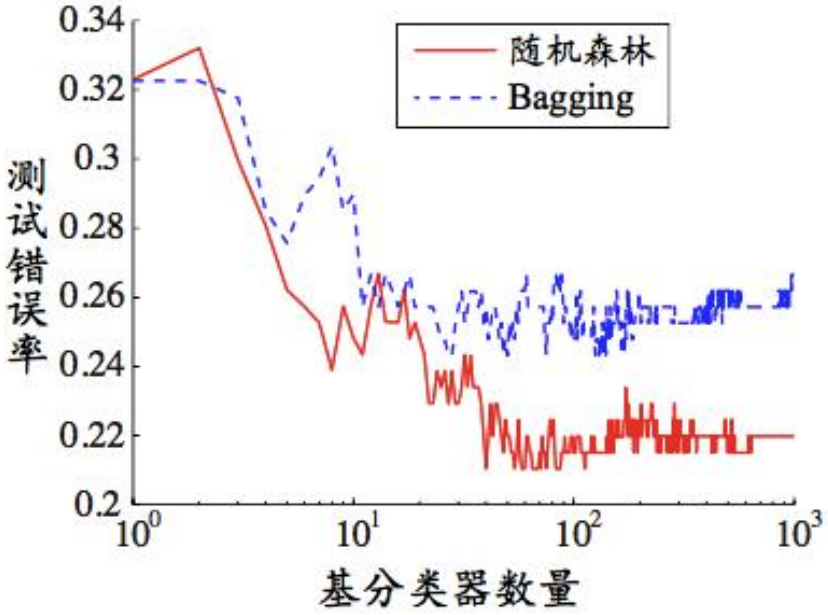
Input: Data set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
Feature subset size K.

Process:

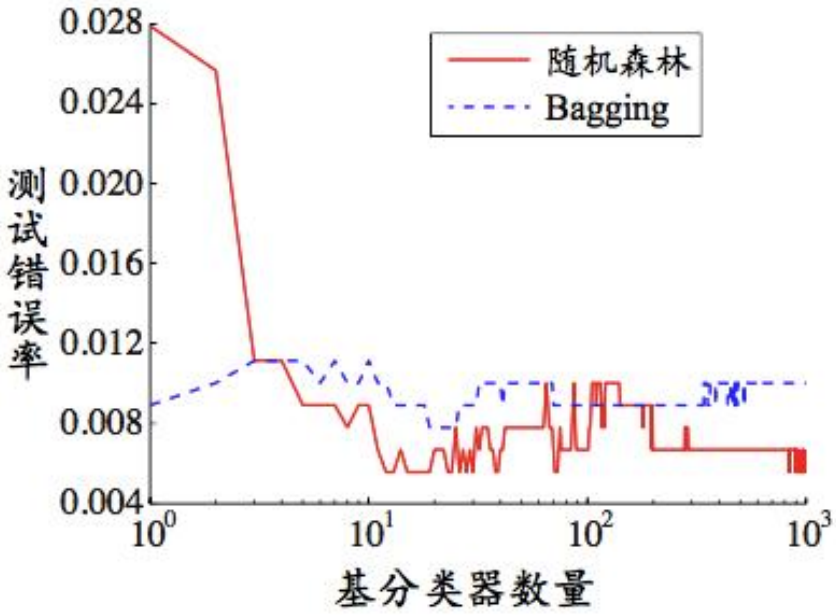
1. $N \leftarrow$ create a tree node based on D;
2. **if** *all instances in the same class* **then return** N
3. $\mathcal{F} \leftarrow$ the set of features that can be split further;
4. **if** $\mathcal{F}$ *is empty* **then return** N
5. $\tilde{\mathcal{F}} \leftarrow$ select K features from $\mathcal{F}$ randomly;
6. $N.f \leftarrow$ the feature which has the best split point in $\tilde{\mathcal{F}}$;
7. $N.p \leftarrow$ the best split point on $N.f$;
8. $D_l \leftarrow$ subset of D with values on $N.f$ smaller than $N.p$;
9. $D_r \leftarrow$ subset of D with values on $N.f$ no smaller than $N.p$;
10. $N_l \leftarrow$ call the process with parameters (D_l, K);
11. $N_r \leftarrow$ call the process with parameters (D_r, K);
12. **return** N

Output: A random decision tree

随机森林实验



(a) glass 数据集



(b) auto-mpg 数据集

在两个 UCI 数据上, 集成规模对随机森林与 Bagging 的影响

随机森林实验

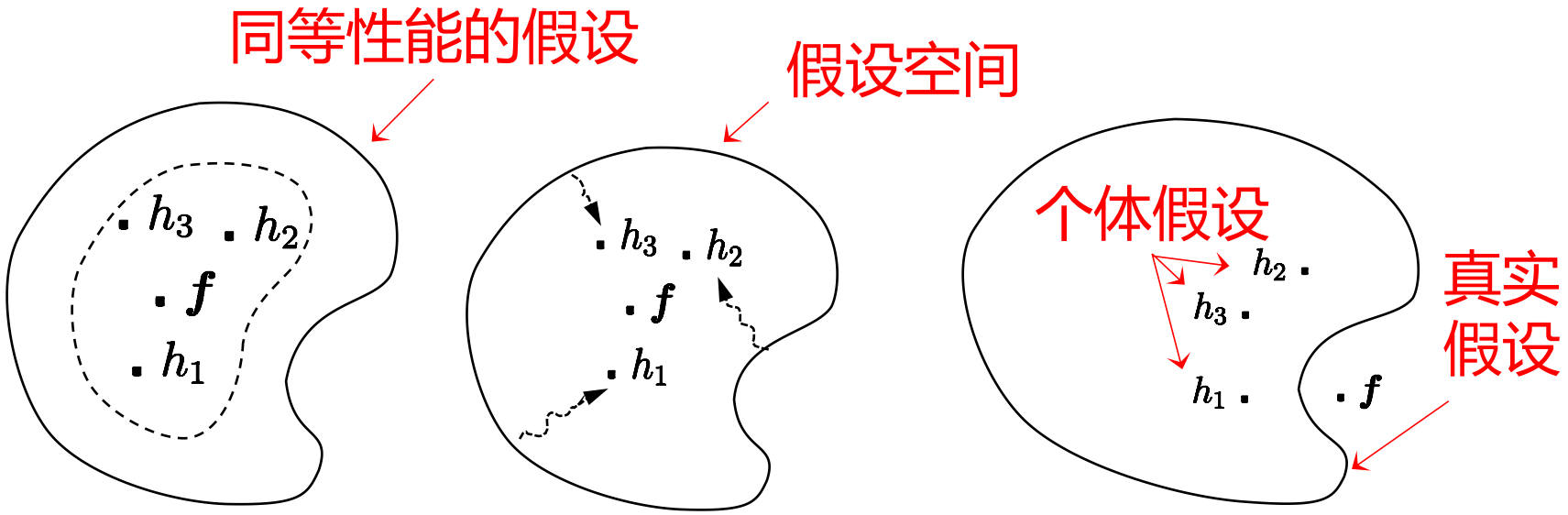
随机森林的起始性能往往相对较差, 特别是在集成中只包含一个基学习器时. 这很容易理解, 因为通过引入属性扰动, 随机森林中个体学习器的性能往往有所降低. 然而, **随着个体学习器数目的增加, 随机森林通常会收敛到更低的泛化误差.**

Part 4

结合策略

结合策略

- 学习器的组合可以从三个方面带来好处：



(a) 统计的原因　(b) 计算的原因　(c) 表示的原因

结合策略 – 平均法

- 简单平均法

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

- 加权平均法

$$H(\boldsymbol{x}) = \sum_{i=1}^{T} w_i h_i(\boldsymbol{x}), \quad w_i \geq 0 \ \text{and} \ \sum_{i=1}^{T} w_i = 1. \tag{1}$$

结合策略 – 平均法

- 简单平均法是加权平均法的特例
- 加权平均法在二十世纪五十年代被广泛使用
- 集成学习中的各种结合方法都可以看成是加权平均法的变种或特例
- 加权平均法可认为是集成学习研究的基本出发点
- 加权平均法未必一定优于简单平均法

结合策略 – 投票法

- 绝对多数投票法（majority voting）

$$H(\boldsymbol{x}) = \begin{cases} c_j & \text{if } \sum_{i=1}^{T} h_i^j(\boldsymbol{x}) > \frac{1}{2} \sum_{k=1}^{l} \sum_{i=1}^{T} h_i^k(\boldsymbol{x}) \\ \text{rejection} & \text{otherwise .} \end{cases}$$

- 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）

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

结合策略 – 学习法

- Stacking是学习法的典型代表

Input: Data set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
First-level learning algorithms $\mathfrak{L}_1, \ldots, \mathfrak{L}_T$;
Second-level learning algorithm $\mathfrak{L}$.

Process:

```
1.  for t = 1,...,T:     % Train a first-level learner by applying the
2.    h_t = L_t(D);      % first-level learning algorithm L_t
3.  end
4.  D' = ∅;              % Generate a new data set
5.  for i = 1,...,m:
6.    for t = 1,...,T:
7.      z_it = h_t(x_i);
8.    end
9.    D' = D' ∪ ((z_i1,...,z_iT), y_i);
10. end
11. h' = L(D');          % Train the second-level learner h' by applying
                         % the second-level learning algorithm L to the
                         % new data set D'.
```

Output: $H(\boldsymbol{x}) = h'(h_1(\boldsymbol{x}), \ldots, h_T(\boldsymbol{x}))$

多响应线性回归(MLR)作为次级学习器的学习算法效果较好

- 贝叶斯模型平均(BMA)

Part 5

多样性

多样性 – 误差-分歧分解

- 假定我们用个体学习器 $h_1,h_2,\ldots,h_T$ 通过加权平均法(1)结合产生的集成来完成回归学习任务 $f:R^d\mapsto R$. 对示例 x, 定义学习器 h_i 的 “分歧" (ambiguity) 为

$$A(h_i \mid \boldsymbol{x}) = \left(h_i(\boldsymbol{x}) - H(\boldsymbol{x})\right)^2$$

- 集成的分歧:

$$\begin{aligned}\overline{A}(h \mid \boldsymbol{x}) &= \sum_{i=1}^{T} w_i A(h_i \mid \boldsymbol{x}) \\ &= \sum_{i=1}^{T} w_i \left(h_i\left(\boldsymbol{x}\right) - H\left(\boldsymbol{x}\right)\right)^2\end{aligned}$$

多样性 – 误差-分歧分解

- 这里的“分歧”项表征了个体学习器在样本 x 上的不一致性, 即在一定程度上反映了个体学习器的多样性. 个体学习器 h_i 和集成 H 的平方误差分别为

$$E(h_i \mid \boldsymbol{x}) = \big(f(\boldsymbol{x}) - h_i(\boldsymbol{x})\big)^2$$

$$E(H \mid \boldsymbol{x}) = \big(f(\boldsymbol{x}) - H(\boldsymbol{x})\big)^2$$

多样性 – 误差-分歧分解

- 令 $\overline{E}(h \mid \boldsymbol{x}) = \sum_{i=1}^{T} w_i \cdot E(h_i \mid \boldsymbol{x})$ 表示个体学习器误差的加权均值，有

$$\begin{aligned}\overline{A}(h \mid \boldsymbol{x}) &= \sum_{i=1}^{T} w_i E(h_i \mid \boldsymbol{x}) - E(H \mid \boldsymbol{x}) \\ &= \overline{E}(h \mid \boldsymbol{x}) - E(H \mid \boldsymbol{x}) \ .\end{aligned}$$

- 上式对所有样本 x 均成立, 令 p(x) 表示样本的概率密度, 则在全样本上有

$$\sum_{i=1}^{T} w_i \int A(h_i \mid \boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x} = \sum_{i=1}^{T} w_i \int E(h_i \mid \boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x} - \int E(H \mid \boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x}$$

多样性 – 误差-分歧分解

- 个体学习器 h_i 在全样本上的泛化误差和分歧项分别为

$$E_i = \int E(h_i \mid \boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x}$$

$$A_i = \int A(h_i \mid \boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x}$$

- 集成的泛化误差为：

$$E = \int E(H \mid \boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x}$$

- 令 $\overline{E} = \sum_{i=1}^{T} w_i E_i$ 表示个体学习器泛化误差的加权均值，$\overline{A} = \sum_{i=1}^{T} w_i A_i$ 表示个体学习器的加权分歧值，有

$$E = \overline{E} - \overline{A}$$

多样性 – 误差-分歧分解

- 这个式子明确提示出: **个体学习器准确性越高、多样性越大, 则集成越好**, 称为“误差—分歧分解 “
- 为什么不能把 $E = \overline{E} - \overline{A}$ 作为优化目标来求解?
  - 在现实任务中很难直接对 $E = \overline{E} - \overline{A}$ 进行优化

    它们定义在整个样本空间上

    $\overline{A}$ 不是一个可直接操作的多样性度量

    上面的推导过程只适用于**回归学习**, 难以直接推广到分类学习任务上去

多样性 – 多样性度量

- **多样性度量**(diversity measure)用于**度量集成中个体学习器的多样性**即估算个体学习器的多样化程度.
- 典型做法是考虑个体分类器的两两相似/不相似性.

多样性 – 多样性度量

- 给定数据集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$，对于二分类问题，$y_i \in \{-1, +1\}$，分类器$h_i$与$h_j$的预测结果联立表为：

	$h_i = +1$	$h_i = -1$
$h_j = +1$	a	c
$h_j = -1$	b	d

$$a + b + c + d = m$$

多样性 – 多样性度量

常见的多样性度量

- 不合度量(Disagreement Measure)

$$dis_{ij} = \frac{b+c}{m}$$

- 相关系数(Correlation Coefficient)

$$\rho_{ij} = \frac{ad-bc}{\sqrt{(a+b)(a+c)(c+d)(b+d)}}$$

多样性 – 多样性度量

常见的多样性度量

- Q-统计量(Q-Statistic)

$$Q_{ij} = \frac{ad - bc}{ad + bc} \qquad |Q_{ij}| \leq |\rho_{ij}|$$

- κ-统计量(Kappa-Statistic)

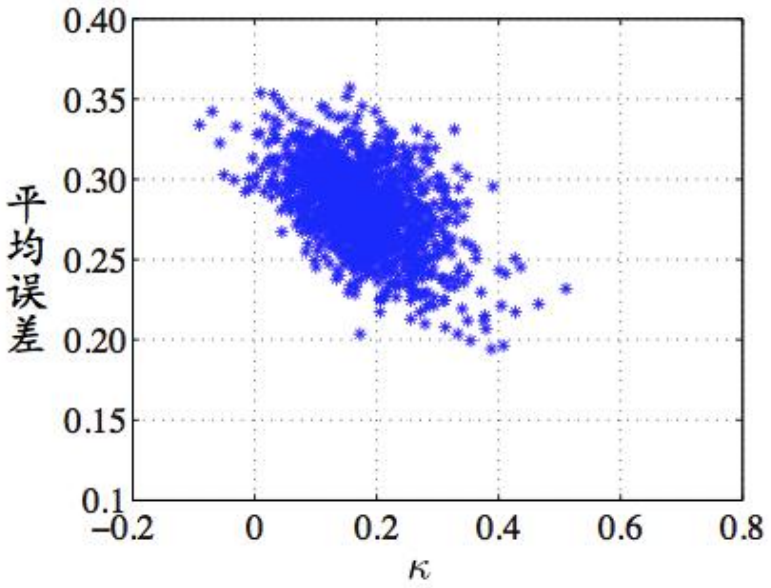
$$\kappa = \frac{p_1 - p_2}{1 - p_2}$$

$$p_1 = \frac{a + d}{m},$$

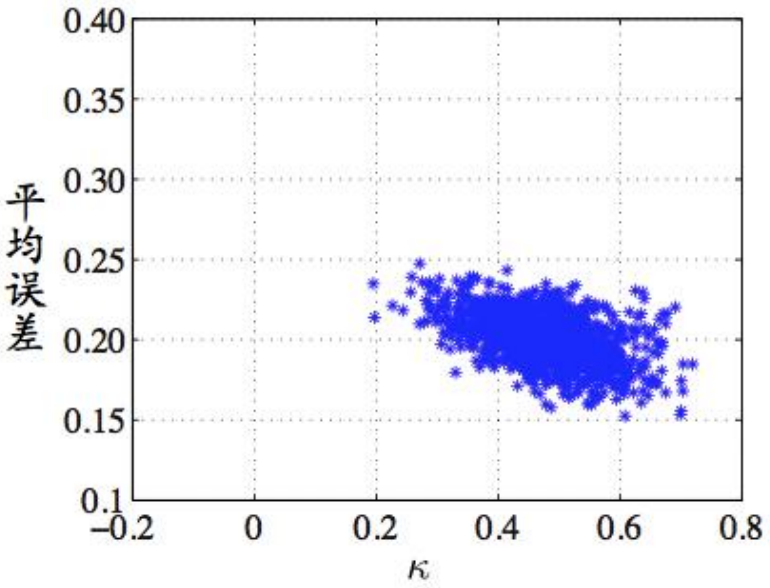
$$p_2 = \frac{(a+b)(a+c) + (c+d)(b+d)}{m^2}$$

多样性 – 多样性度量

“κ-误差图”：将每一对分类器作为图上的一 个点, 横坐标是这对分类器的 κ 值, 纵坐标是它们的平均误差



(a) AdaBoost 集成



(b) Bagging 集成

多样性 – 多样性增强

- 常见的增强个体学习器的多样性的方法
  - 数据样本扰动
  - 输入属性扰动
  - 输出表示扰动
  - 算法参数扰动

多样性 – 多样性增强 - 数据样本扰动

- 数据样本扰动通常是基于采样法
  - Bagging中的自助采样法
  - Adaboost中的序列采样

数据样本扰动对“不稳定基学习器”很有效

- 对数据样本的扰动敏感的基学习器(不稳定基学习器)
  - 决策树，神经网络等
- 对数据样本的扰动不敏感的基学习器(稳定基学习器)
  - 线性学习器，支持向量机，朴素贝叶斯，k近邻等

多样性 – 多样性增强 – 输入属性扰动

- 随机子空间算法(random subspace)

输入: 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$;
基学习算法 $\mathfrak{L}$;
基学习器数 T;
子空间属性数 d'.

过程:

1: **for** $t = 1, 2, \ldots, T$ **do**
2: $\quad \mathcal{F}_t = \mathrm{RS}(D, d')$
3: $\quad D_t = \mathrm{Map}_{\mathcal{F}_t}(D)$
4: $\quad h_t = \mathfrak{L}(D_t)$
5: **end for**

输出: $H(\boldsymbol{x}) = \arg\max_{y \in \mathcal{Y}} \sum_{t=1}^{T} \mathbb{I}\left(h_t\left(\mathrm{Map}_{\mathcal{F}_t}(\boldsymbol{x})\right) = y\right)$

多样性 – 多样性增强 – 输出表示扰动

- 翻转法(Flipping Output)
- 输出调剂法(Output Smearing)
- ECOC法

多样性 – 多样性增强 – 算法参数扰动

- 负相关法
- 不同的多样性增强机制同时使用

THANK YOU