第八章 集成学习

§8.1 背景知识

§8.2 学习策略

§8.1 背景知识

一、问题引入

二、个体与集成

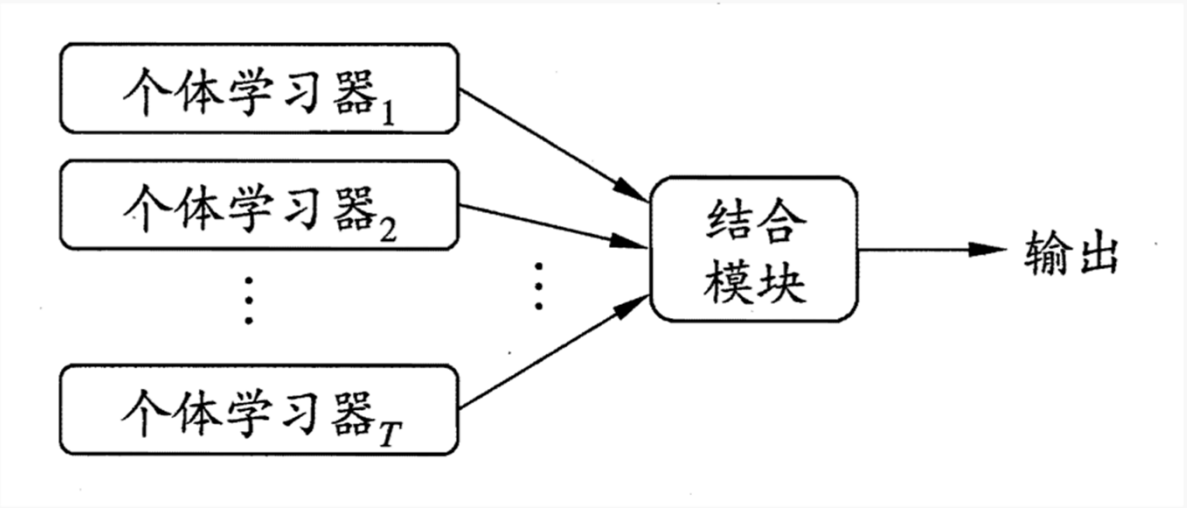
问题引入

- 结合多个模型取得更好的效果
  - 一种直观提升模式识别系统性能的方式：“三个臭皮匠，顶个诸葛亮”

个体与集成

集成学习

- **定义：**通过构建并结合多个学习器完成学习任务
- **流程：**按照某种策略训练一组“个体学习器”，再通过某种策略将其结合



- 同质集成与异质集成
- 不同集成学习方法中，个体学习器的**训练方式**和**结合策略**各有不同

个体与集成

集成学习

- 好坏不同的东西结合在一起，最终的性能一般介于最好与最坏之间
- **问题：**如何保证集成学习的有效性？

	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

- 要获得好的集成效果，个体学习器应 **“好而不同”**
  - 具有一定准确性
  - 学习器之间具有差异

个体与集成

Hoeffding（霍夫丁）不等式

- 考虑伯努利实验，该实验是在同样条件下重复地、相互独立地进行，其特点是该随机试验只有两种可能结果：发生或者不发生
- 比较熟悉的一种就是0-1分布，也叫两点分布
- 用$h(n)$表示抛n次硬币正面向上的概率，则正面向上的次数不超过k的概率为

$$P(H(n) \leq k) = \sum_{i=0}^{k} \binom{n}{i} p^i (1-p)^{n-i}$$

- 例子：从一个装有绿色和黄色的罐子里随机地摸出小球，根据摸出小球的概率来估计整个罐子中绿色球和黄色球的所占的比例
- 当抽出的样本数越多，最终预测出的绿球占整个罐子中小球的概率会越趋近于实际罐子中绿色小球占整个罐子小球的概率

个体与集成

简单理论分析

- 考虑**二分类问题**：$y \in \{-1, +1\}$和真实函数f，假定**每个基分类器**h_i错误率为ϵ：

$$P\big(h_i(x) \neq f(x)\big) = \epsilon$$

- 通过投票法结合T个基分类器，若有**超过半数正确**，则集成分类正确：

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

用于限定有界随机变量和过大或过小的概率

- 假设基分类器的错误率相互独立，根据Hoeffding（霍夫丁）不等式，集成的错误率为：

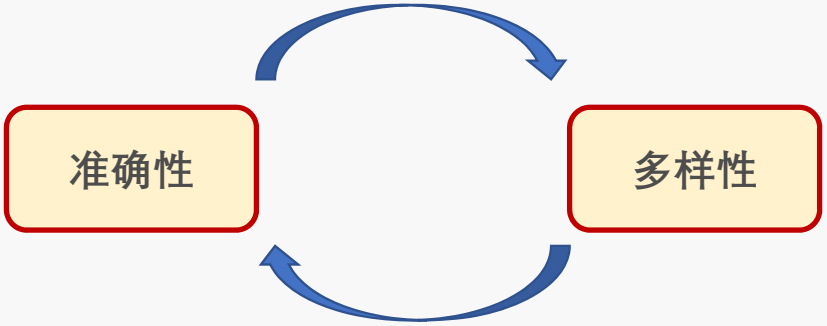
$$P(H(x) \neq f(x)) = \sum_{k=0}^{\lfloor T/2 \rfloor} \binom{T}{k} (1-\epsilon)^k \epsilon^{T-k} \leq \exp\left(-\frac{1}{2} T (1-2\epsilon)^2\right)$$

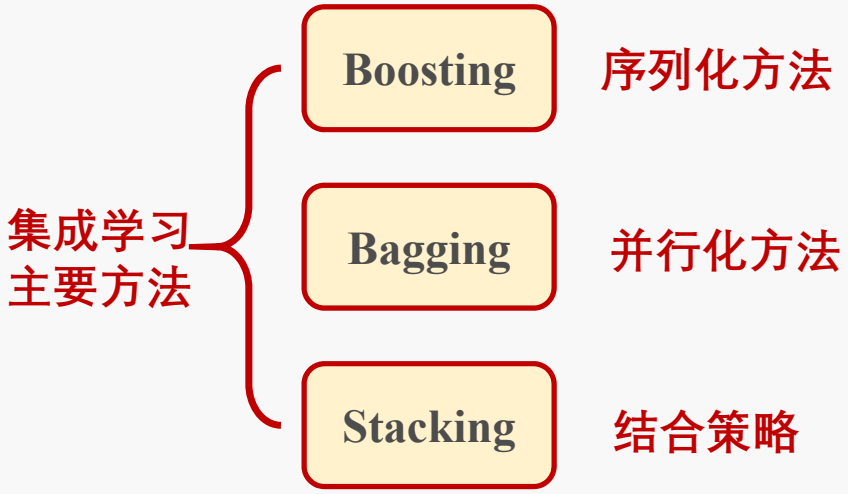
- 因此随着T的增大，集成的错误率以指数级趋近于0

个体与集成

简单理论分析

- 上述分析中存在的**关键假设**：基学习器的误差相互独立，**一般无法满足**
- 现实任务中，个体学习器是为了解决同一个问题而训练的，不可能相互独立
- 个体学习器的**准确性**和**多样性**存在冲突
- 集成学习研究的核心：训练并结合“好而不同”的个体学习器





§8.2 学习策略

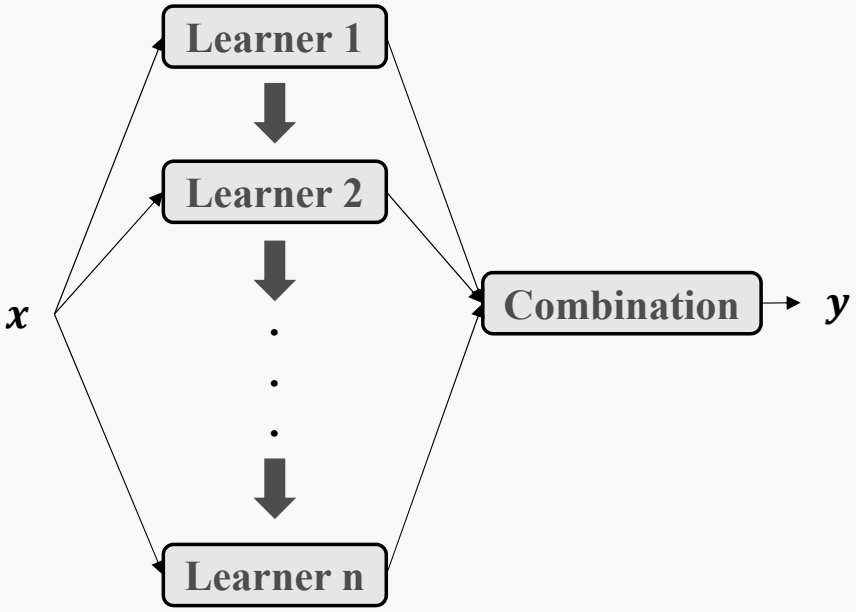
一、Boosting

二、Bagging

三、Stacking

Boosting

- 序列化方法：Boosting
  - 串行式训练基学习器
  - 根据先前基学习器的表现调整训练样本
  - 新增的基学习器关注先前基学习器预测错误的部分
  - **主要思想**：结合若干弱学习器以避免欠拟合（降低偏差）
  - **代表方法**：Adaboost、提升树



Boosting

Adaboost

- 算法如下图所示，其中$y_i \in \{-1, +1\}$，f是真实函数

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: $h_t = \mathfrak{L}(D, \mathcal{D}_t)$; ← 训练分类器
4: $\epsilon_t = P_{\boldsymbol{x} \sim \mathcal{D}_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$;
5: **if** $\epsilon_t > 0.5$ **then break**
6: $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$; ← 分类器权重
7: $\mathcal{D}_{t+1}(\boldsymbol{x}) = \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t} \times \begin{cases} \exp(-\alpha_t), & \text{if } h_t(\boldsymbol{x}) = f(\boldsymbol{x}) \\ \exp(\alpha_t), & \text{if } h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}) \end{cases}$ ← 更新分布
$= \frac{\mathcal{D}_t(\boldsymbol{x}) \exp(-\alpha_t f(\boldsymbol{x}) h_t(\boldsymbol{x}))}{Z_t}$
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：替代损失函数
  - 基于“加性模型”的推导
  - 通过基学习器的线性组合

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

  - 最小化指数损失函数

$$l_{exp}(H|\mathbb{D}) = \mathbb{E}_{x\sim\mathbb{D}}[e^{-f(x)H(x)}]$$

  - 若$H(x)$能令指数损失函数最小化，则考虑上式对$H(x)$的偏导

$$\frac{\partial l_{exp}(H|\mathbb{D})}{\partial H(x)} = \frac{\partial[P(f(x)=1|x)e^{-H(x)} + P(f(x)=-1|x)e^{H(x)}]}{\partial H(x)}$$

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

Boosting

- Adaboost推导：替代损失函数
  - 令偏导为零，解得

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

  - 因此有

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

  - $\operatorname{sign}\big(H(x)\big)$达到了贝叶斯最优错误率，可作为该二分类任务的替代损失函数

Boosting

Adaboost推导：分类器权重

- 第一个基分类器h_1是基学习算法直接用于初始数据分布得到
- 此后迭代生成h_t和α_t，当基分类器h_t基于分布$\mathbb{D}_t$产生后，其权重α_t应该使得$\alpha_t h_t$最小化当前数据分布下的指数损失函数

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

- 其中$\epsilon_t = P_{x\sim\mathbb{D}_t}(h_t(x)\neq f(x))$，考虑导数

$$\frac{\partial l_{exp}(\alpha_t h_t|\mathbb{D}_t)}{\partial\alpha_t} = -e^{-\alpha_t}(1-\epsilon_t)+e^{\alpha_t}\epsilon_t$$

- 令导数为零，可得分类器权重更新公式：$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$

Boosting

Adaboost推导：更新分布

- 在获得H_{t-1}（求解出α_{t-1}）之后，需要对样本分布进行调整（更新$\mathbb{D}_{t-1}$）
- 在调整后样本分布上训练出来的基学习器h_t，应纠正H_{t-1}的错误，理想情况下h_t应最小化

$$\begin{aligned} l_{exp}(H_{t-1}+h_t|\mathbb{D}) &= \mathbb{E}_{x\sim\mathbb{D}}\left[e^{-f(x)(H_{t-1}(x)+h_t(x))}\right] \\ &= \mathbb{E}_{x\sim\mathbb{D}}\left[e^{-f(x)H_{t-1}(x)}e^{-f(x)h_t(x)}\right] \end{aligned}$$

- 注意到$f^2(x)=h_t^2(x)=1$，对$e^{-f(x)h_t(x)}$进行泰勒展开近似为

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

Boosting

Adaboost推导：更新分布

- 理想的基学习器满足

$$h_t(x) = \underset{h}{\operatorname{argmin}}\, l_{exp}(H_{t-1} + h|\mathcal{D})$$

$$= \underset{h}{\operatorname{argmin}}\, \mathbb{E}_{x\sim\mathcal{D}}\left[e^{-f(x)H_{t-1}(x)}\left(1 - f(x)h(x) + \frac{1}{2}\right)\right]$$

$$= \underset{h}{\operatorname{argmax}}\, \mathbb{E}_{x\sim\mathcal{D}}\left[e^{-f(x)H_{t-1}(x)}f(x)h(x)\right]$$

$$= \underset{h}{\operatorname{argmax}}\, \mathbb{E}_{x\sim\mathcal{D}}\left[\frac{e^{-f(x)H_{t-1}(x)}}{\mathbb{E}_{x\sim\mathcal{D}}\left[e^{-f(x)H_{t-1}(x)}\right]}f(x)h(x)\right]$$

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

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

Boosting

- **Adaboost推导：更新分布**
  - 因此有如下等价关系

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

$$= \underset{h}{\operatorname{argmax}} \mathbb{E}_{x\sim\mathcal{D}_t}[f(x)h(x)]$$

  - 由$f(x), h(x) \in \{-1, +1\}$，有

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

  - 因此

$$h_t(x) = \underset{h}{\operatorname{argmin}} \mathbb{E}_{x\sim\mathcal{D}_t}[\mathbb{I}(f(x) \neq h(x))]$$

  - 说明理想的基学习器h_t将在分布$\mathcal{D}_t$下最小化分类误差，因此$\mathcal{D}_t$即为我们所需要的调整后的数据分布

Boosting

Adaboost推导：更新分布

- 考虑到D_t和D_{t+1}的关系，有

$$\mathrm{D}_{t+1}(x) = \frac{\mathrm{D}(x)e^{-f(x)H_t(x)}}{\mathbb{E}_{x\sim\mathrm{D}}[e^{-f(x)H_t(x)}]}$$

$$= \frac{\mathrm{D}(x)e^{-f(x)H_{t-1}(x)-f(x)\alpha_t h_t(x)}}{\mathbb{E}_{x\sim\mathrm{D}}[e^{-f(x)H_t(x)}]}$$

$$= \mathrm{D}_t(x)\cdot e^{-f(x)\alpha_t h_t(x)}\frac{\mathbb{E}_{x\sim\mathrm{D}}[e^{-f(x)H_{t-1}(x)}]}{\mathbb{E}_{x\sim\mathrm{D}}[e^{-f(x)H_t(x)}]}$$

- 这正是数据分布的更新公式

Boosting

Adaboost

- **例题：**训练数据如下表所示，假设弱分类器由$x < v$或$x > v$产生，其阈值v使该分类器在训练数据集上分类误差率最低。试用AdaBoost算法学习一个强分类器

序号	1	2	3	4	5	6	7	8	9	10
x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	−1	−1	−1	1	1	1	−1

- **解答：**首先初始化数据权值分布

$$\mathbb{D}_1 = (0.1, 0.1, \ldots, 0.1)$$

- 对$t = 1$
- 在权值分布为$\mathbb{D}_1$的训练数据上，阈值v取2.5时分类误差率最低，故基本分类器为

$$h_1(x) = \begin{cases} 1, x < 2.5 \\ -1, x > 2.5 \end{cases}$$

Boosting

Adaboost

- $h_1(x)$在训练数据集上的误差率$\epsilon_1 = 0.3$
- 计算$h_1(x)$的系数：

$$\alpha_1 = \frac{1}{2}\ln\left(\frac{1-\epsilon_1}{\epsilon_1}\right) = 0.4236$$

- 更新训练数据的权值分布：

$$\mathbb{D}_2(x) = \frac{\mathbb{D}_1(x)e^{-f(x)\alpha_1 h_1(x)}}{Z_1}$$

- $\mathbb{D}_2 = (0.07143, 0.07143, 0.07143, 0.07143, 0.07143, 0.07143, 0.16667, 0.16667, 0.16667, 0.07143)$
- 组合分类器为$H(x) = \text{sign}\left(0.4236h_1(x)\right)$，在训练集上有3个错误分类点
- 对$t = 2$
- 在权值分布为$\mathbb{D}_2$的训练数据上，阈值v取8.5时分类误差率最低

Boosting

- Adaboost
  - 在权值分布为$\mathbb{D}_2$的训练数据上，阈值v取8.5时分类误差率最低，得到$h_2(x)$
  - 中间过程跳过，得到$\alpha_2 = 0.6496$和新的数据分布$\mathbb{D}_3$
  - 组合分类器为$H(x) = \text{sign}\left(0.4236h_1(x) + 0.6496h_2(x)\right)$，在训练集上有3个误分类点
  - 对$t = 3$
  - 过程省略
  - 组合分类器为$H(x) = \text{sign}\left(0.4236h_1(x) + 0.6496h_2(x) + 0.7514h_3(x)\right)$，在训练集上有0个误分类点
  - 以此作为最终分类器

Boosting

提升树

- 以分类树或回归树为基学习器的boosting方法
- 当决策树是由一个根结点直接连接两个叶节点时，称为决策树桩
- 对于二分类问题，提升树算法只需要将Adaboost的基学习器限制为二分类树即可，属于Adaboost算法的特殊情况；对于回归问题：

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

输出：提升树 $f_M(x)$。

(1) 初始化 $f_0(x)=0$。

(2) 对 $m=1,2,\cdots,M$。

(a) 按式 (8.27) 计算残差：

$$r_{mi}=y_i-f_{m-1}(x_i),\quad i=1,2,\cdots,N$$

例如使用平方误差损失函数 ←

(b) 拟合残差 r_{mi} 学习一个回归树，得到 $T(x;\Theta_m)$。

(c) 更新 $f_m(x)=f_{m-1}(x)+T(x;\Theta_m)$。

(3) 得到回归问题提升树

$$f_M(x)=\sum_{m=1}^{M}T(x;\Theta_m)$$

■

Boosting

提升树

- **例题：**训练数据如下表所示，只用树桩作为基函数，学习该回归问题的提升树模型

x_i	1	2	3	4	5	6	7	8	9	10
y_i	5.56	5.70	5.91	6.40	6.80	7.05	8.90	8.70	9.00	9.05

- **解答：**首先求$f_1(x)$，即回归树$T_1(x)$
- 通过以下优化问题

$$\min_{s}\left[\min_{c_1}\sum_{x_i\in R_1}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2}(y_i-c_2)^2\right]$$

- 求解训练数据的切分点s

$$R_1=\{x|x\le s\},R_2=\{x|x>s\}$$

Boosting

提升树

- 求得在R_1, R_2内部使平方损失误差达到最小值的c_1, c_2为

$$c_1 = \frac{1}{N_1}\sum_{x_i \in R_1} y_i \, , c_2 = \frac{1}{N_2}\sum_{x_i \in R_2} y_i$$

- 这里N_1, N_2是R_1, R_2的样本点数
- 考虑如下切分点：$1.5, 2.5, 3.5, 4.5, \ldots, 8.5, 9.5$。令

$$m(s) = \min_{c_1} \sum_{x_i \in R_1} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2} (y_i - c_2)^2$$

- 则切分点s与$m(s)$的对应表如下

s	1.5	2.5	3.5	4.5	5.5	6.5	7.5	8.5	9.5
$m(s)$	15.72	12.07	8.36	5.78	3.91	1.93	8.01	11.73	15.74

- 当$s = 6.5$时$m(s)$达到最小值，此时$c_1 = 6.24, c_2 = 8.91$

Boosting

提升树

- 因此回归树

$$f_1(x) = T_1(x) = \begin{cases} 6.24, x < 6.5 \\ 8.91, x \geq 6.5 \end{cases}$$

- 用$f_1(x)$拟合后，训练数据的残差如下

x_i	1	2	3	4	5	6	7	8	9	10
r_{2i}	−0.68	−0.54	−0.33	0.16	0.56	0.81	−0.01	−0.21	0.09	0.14

- 第二步求解$T_2(x)$，与求解$T_1(x)$类似，只是拟合的数据是残差
- 以此类推，直到误差满足要求

Boosting

梯度提升树

- 当损失函数是平方损失和指数损失函数时，优化较为简单
- 对于一般函数，每一步优化存在困难，可利用损失函数的负梯度在当前模型的值作为残差的近似值，用于拟合回归树

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

（a）对 $i=1,2,\cdots,N$，计算

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

对于平方损失函数，就是残差；
对于其他损失函数，是残差的近似值

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

（c）对 $j=1,2,\cdots,J$，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 $f_m(x)=f_{m-1}(x)+\sum_{j=1}^{J}c_{mj}I(x\in R_{mj})$

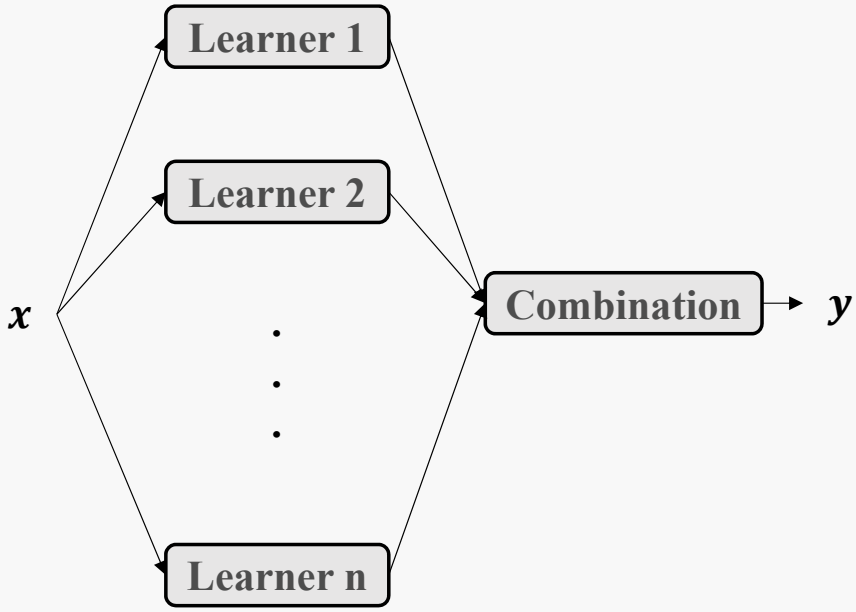
（3）得到回归树

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

■

Bagging

- 并行化方法：Bagging
  - 并行式训练基学习器
  - 基学习器应尽可能相互独立
  - 通过Bootstrap采样生成若干子训练集，训练出具有差异的基学习器进行组合
  - **主要思想**：结合若干强学习器以避免过拟合（降低方差）
  - **代表方法**：Bootstrap AGGregatING、随机森林



Bagging

Bootstrap AGGregatING

- 泛化性能强的集成：个体学习器之间应相互独立
- “独立”在现实任务无法做到，但可以**使基学习器尽可能具有较大差异**
- 可通过采样法产生若干不同子集，通过训练数据的不同，保证基学习器的差异

差异性 → 采样不同的训练集

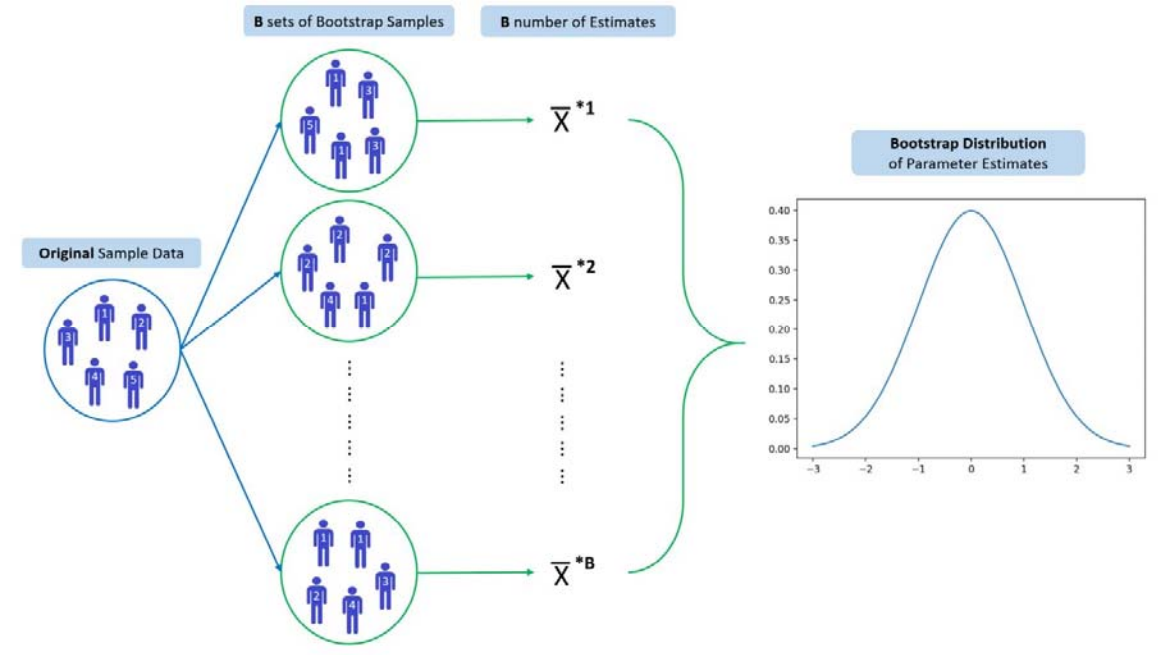
- 为保证集成的性能：个体学习器不能太差
- 若采样出的各个子集完全不同，则每个基学习器的训练数据过少，甚至不足以进行有效学习
- 可考虑使用相互有交叠的采样子集

准确性 → 各个子集之间应有交叠

Bagging

Bootstrap AGGregatING

- 自助采样法（Bootstrap Sampling）：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，再把该样本放回初始数据集，这样经过m次随机采样操作，可得到含m个样本的采样集



Bagging

- **Bootstrap AGGregatING**
  - 假设某样本在m次采样中始终不被采到
  - 由于放回采样，每次不被采到的概率均为$1-\frac{1}{m}$
  - 该样本m次均不被采样到的概率趋近于

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

  - 即某个样本始终不被采样到的概率为36.8%，因此约有63.2%的样本出现在采样集中
  - 通过自助采样法，我们可以采样出T个含m个训练样本的采样集，并基于每个采样集训练出基学习器进行结合
  - 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

Bootstrap AGGregatING

- 自助采样法中，每个基学习器只使用了初始训练集中约63.2%的样本
- 剩下的样本可用作验证集，对泛化性能进行“包外估计”
- 不妨令D_t表示h_t实际使用的训练样本集，令$H^{oob}(x)$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

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

- 则Bagging泛化误差的包外估计为

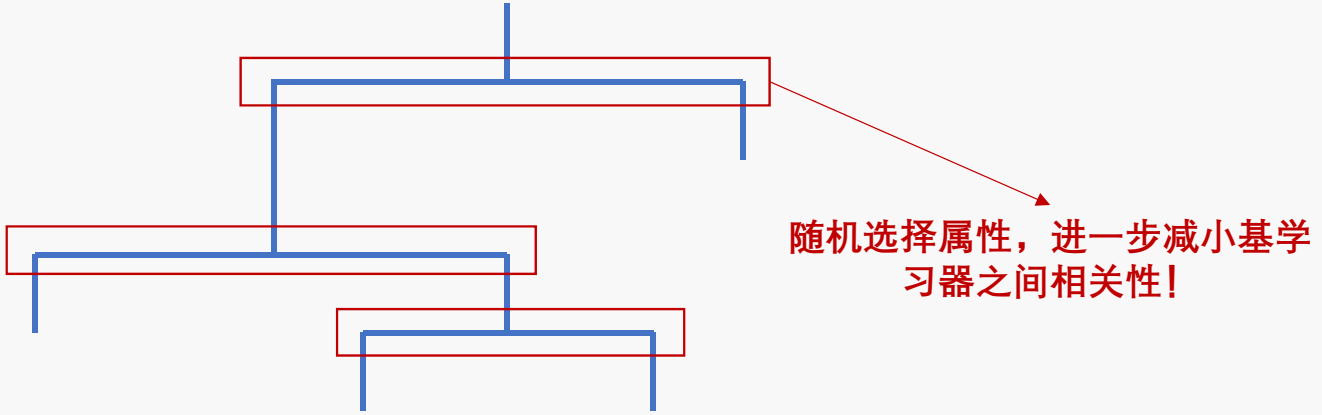
$$\epsilon^{oob} = \frac{1}{|D|} \sum_{(x,y) \in D} \mathbb{I}(H^{oob}(x) \neq y)$$

- 包外样本的其他用途：**辅助决策树进行剪枝，辅助神经网络early stop**等

Bagging

随机森林

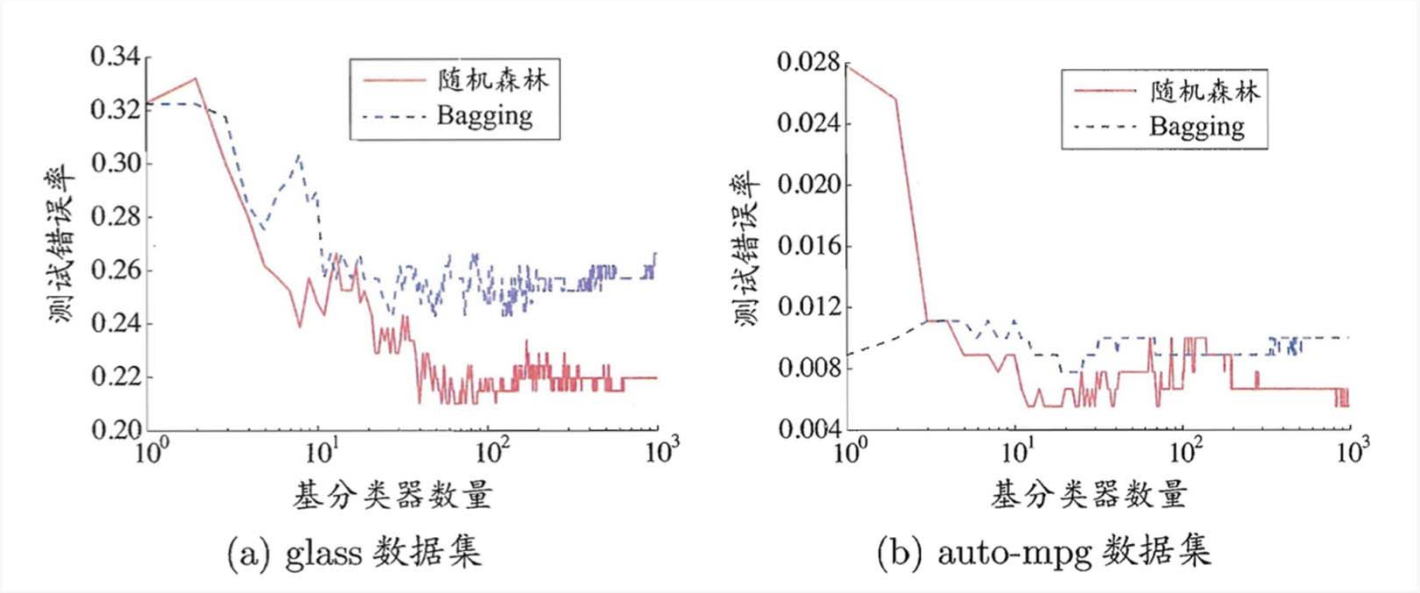
- **动机**：进一步减小基学习器之间的相关性
- 用决策树作为基学习器，修改决策树的生长过程以进一步减小相关性
- 传统决策树：划分属性时选择当前结点的属性集合（d个）中的最优属性
- 随机森林：对每个结点的属性集合随机选一个包含k个属性的子集，在子集中选择最优属性进行划分
- $k = d$时基决策树的构建与传统决策树相同，一般情况推荐值为$k = \log_2 d$



Bagging

随机森林

- 与Bagging相似，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，这是因为随机属性扰动往往会降低个体学习器的性能
- 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，且训练效率也优于Bagging，这是因为随机森林中基决策树需要考察的属性更少



(a) glass 数据集　　(b) auto-mpg 数据集

Bagging

- 偏差-方差分解
  - 之前提到Boosting降低偏差，Bagging降低方差
  - 用偏差-方差分解解释机器学习算法的泛化性
  - 对测试样本x，令y_D为x在数据集中的标记，y为x的真实标记，$f(x;D)$为训练集D上学得模型f在x上的预测输出
  - 以回归任务为例，学习算法的期望预测为

$$\bar{f}(x) = \mathbb{E}_D[f(x;D)]$$

  - 使用样本数相同的不同训练集产生的方差为

$$var(x) = \mathbb{E}_D\left[\left(f(x;D) - \bar{f}(x)\right)^2\right]$$

  - 噪声为$\epsilon^2 = \mathbb{E}_D[(y_D - y)^2]$
  - 期望输出与真实标记的差别称为偏差，即

$$bias^2(x) = \left(\bar{f}(x) - y\right)^2$$

Bagging

- 偏差-方差分解
  - 为便于讨论，假定噪声为零，即$\mathbb{E}_D[y_D - y] = 0$
  - 对算法的期望泛化误差进行分解

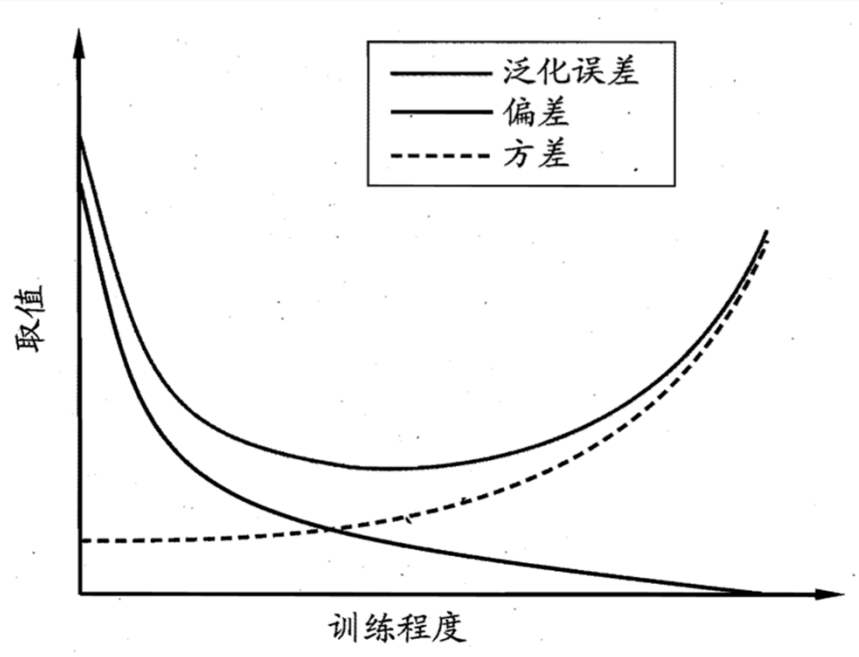
$$
\begin{aligned}
E(f;D) &= \mathbb{E}_D[(f(x;D) - y_D)^2] \\
&= \mathbb{E}_D[(f(x;D) - \bar{f}(x) + \bar{f}(x) - y_D)^2] \\
&= \mathbb{E}_D\left[\left(f(x;D) - \bar{f}(x)\right)^2\right] + \mathbb{E}_D\left[\left(\bar{f}(x) - y_D\right)^2\right] + \mathbb{E}_D[2(f(x;D) - \bar{f}(x))(\bar{f}(x) - y_D)] \\
&= \mathbb{E}_D\left[\left(f(x;D) - \bar{f}(x)\right)^2\right] + \mathbb{E}_D\left[\left(\bar{f}(x) - y_D\right)^2\right] \\
&= \mathbb{E}_D\left[\left(f(x;D) - \bar{f}(x)\right)^2\right] + \mathbb{E}_D\left[\left(\bar{f}(x) - y + y - y_D\right)^2\right] \\
&= \mathbb{E}_D\left[\left(f(x;D) - \bar{f}(x)\right)^2\right] + \mathbb{E}_D\left[\left(\bar{f}(x) - y\right)^2\right] + \mathbb{E}_D[(y - y_D)^2] + \mathbb{E}_D[2(\bar{f}(x) - y)(y - y_D)] \\
&= \mathbb{E}_D\left[\left(f(x;D) - \bar{f}(x)\right)^2\right] + \left(\bar{f}(x) - y\right)^2 + \mathbb{E}_D[(y - y_D)^2]
\end{aligned}
$$

  - $E(f;D) = var(x) + bias^2(x) + \epsilon^2$，即泛化误差是方差、偏差与噪声之和

Bagging

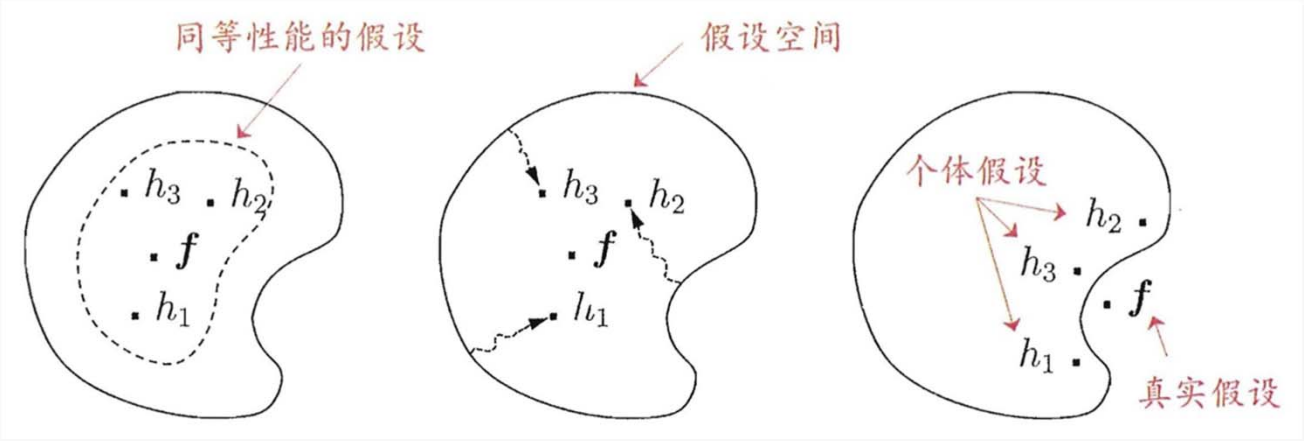
偏差-方差分解

- 偏差度量了学习算法的期望预测与真实结果的偏离程度，刻画了学习算法本身的拟合能力
- 方差度量了同样大小的训练集变动所导致的学习性能的变化，刻画了数据扰动所造成的影响噪声
- 噪声表达了在当前任务上任何学习算法所能达到的期望泛化误差的下界，刻画了学习问题本身的难度
- **偏差-方差窘境**：训练初期，学习器的拟合能力不够强，泛化误差由偏差主导；训练充足后，数据自身的非全局特性被学习器学习到，发生过拟合，由方差主导泛化误差



Stacking

- 学习器结合的三大好处
  - 从**统计方面**看：由于学习任务假设空间大，可能有多个假设在训练集上达到同等性能，若使用单学习器可能因为误选而导致泛化性能不佳
  - 从**计算方面**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，多次运行之后进行结合可降低陷入局部极小点的风险
  - 从**表示方面**看：某些学习任务的真实假设可能不在当前算法的假设空间内，使用单学习器肯定无效，结合多个学习器可扩大假设空间，进行更好的近似



Stacking

- 结合策略
  - 常用策略：平均法、加权平均法、投票法、加权投票法
  - 当训练数据很多时，一种更为强大的结合策略是“学习法”
  - 学习法：通过另一个学习器来进行结合，代表方法为Stacking

输入： 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
初级学习算法 $\mathfrak{L}_1, \mathfrak{L}_2, \ldots, \mathfrak{L}_T$;
次级学习算法 $\mathfrak{L}$.

过程：

1: **for** $t = 1, 2, \ldots, T$ **do**
2: $\quad h_t = \mathfrak{L}_t(D)$;
3: **end for**
4: $D' = \varnothing$;
5: **for** $i = 1, 2, \ldots, m$ **do**
6: $\quad$ **for** $t = 1, 2, \ldots, T$ **do**
7: $\quad\quad z_{it} = h_t(\boldsymbol{x}_i)$;
8: $\quad$ **end for**
9: $\quad D' = D' \cup ((z_{i1}, z_{i2}, \ldots, z_{iT}), y_i)$;
10: **end for**
11: $h' = \mathfrak{L}(D')$;

输出： $H(\boldsymbol{x}) = h'(h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots, h_T(\boldsymbol{x}))$

训练初级学习器

初级学习器的输出作为样例输入特征，生成新数据集

训练次级学习器

Stacking

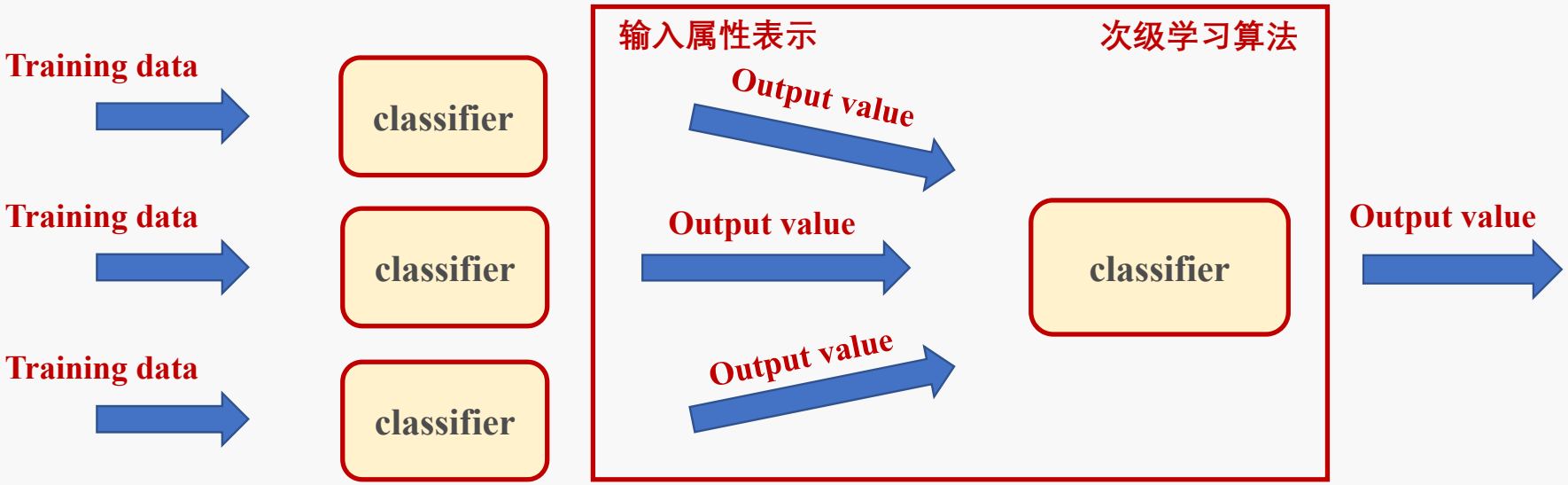
结合策略

- 次级训练集由初级学习器产生，但如果直接用初级学习器的训练集来产生次级训练集，会存在较大过拟合风险；因此，通过交叉验证或留一法，用初级学习器未使用的样本来产生次级学习器的训练样本
- 以k折交叉验证为例，初级训练集D被随机划分为k个大小相似的集合$D_1, D_2, \ldots, D_k$，令D_j和$\overline{D}_j = D \backslash D_j$表示第$j$折的测试集和训练集
- 给定T个初级学习算法，初级学习器$h_t^{(j)}$通过在$\overline{D}_j$上使用第t个学习算法而得
- 对D_j中每个样本x_i，令$z_{it} = h_t^{(j)}(x_i)$，则由x_i所产生的次级训练样例为$z_i = (z_{i1}; z_{i2}; \ldots; z_{iT})$，标记为$y_i$
- 完成交叉验证后，从这T个初级学习器产生的次级训练集是$D' = \{(z_i, y_i)\}_{i=1}^{m}$
- 将D'用于训练次级学习器

Stacking

结合策略

- 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
- 将初级学习器的输出类概率作为次级学习器的输入属性，用多响应线性回归（MLR）作为次级学习算法效果较好，MLR中使用不同的属性集效果更佳

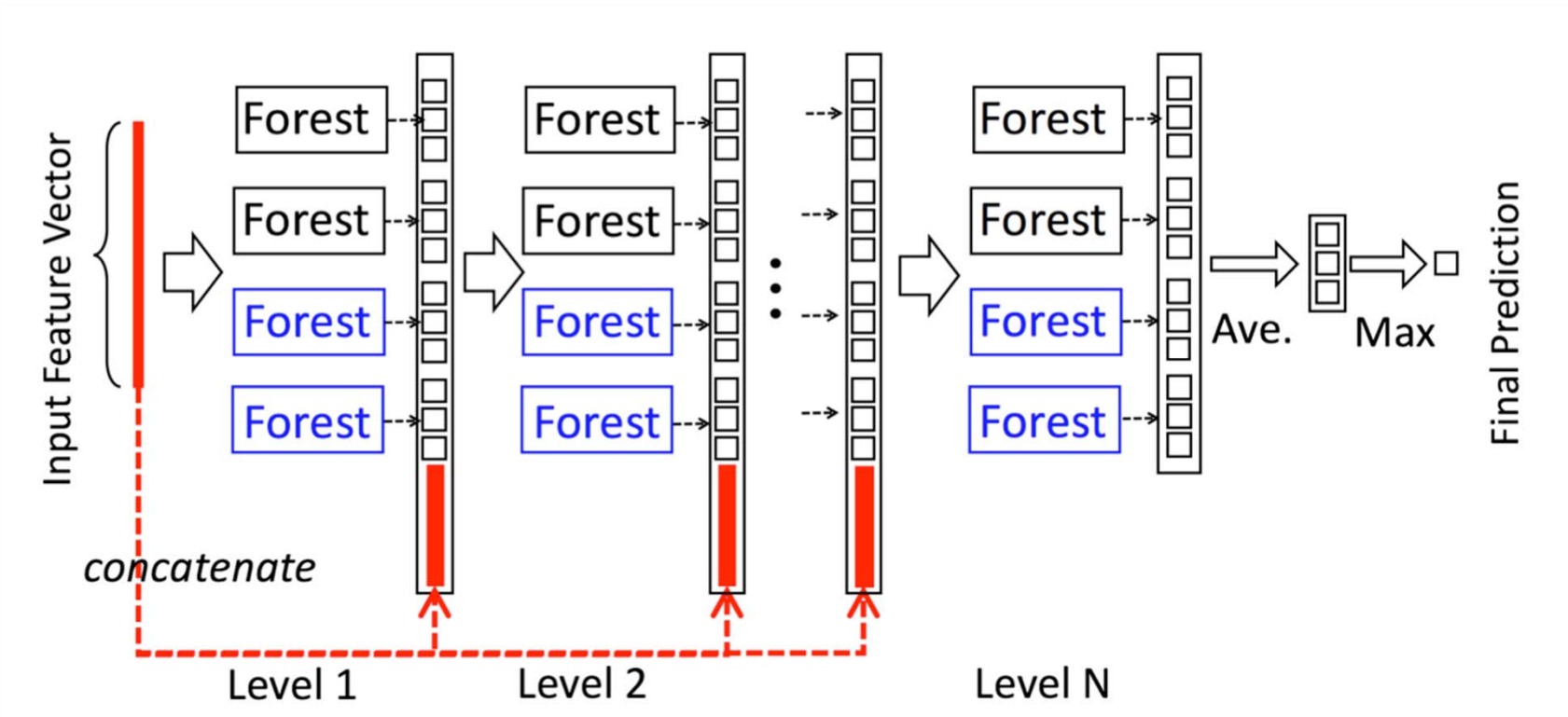


Stacking

- 深度森林
  - 借鉴深度网络成功的经验，设计模型既可以学到足够复杂的特征表示，又避免参数量过大、超参敏感的问题
  - 用不可微的模块（决策树）构建深度模型
  - 通过Stacking技术增强决策树的性能，构建深度森林模型

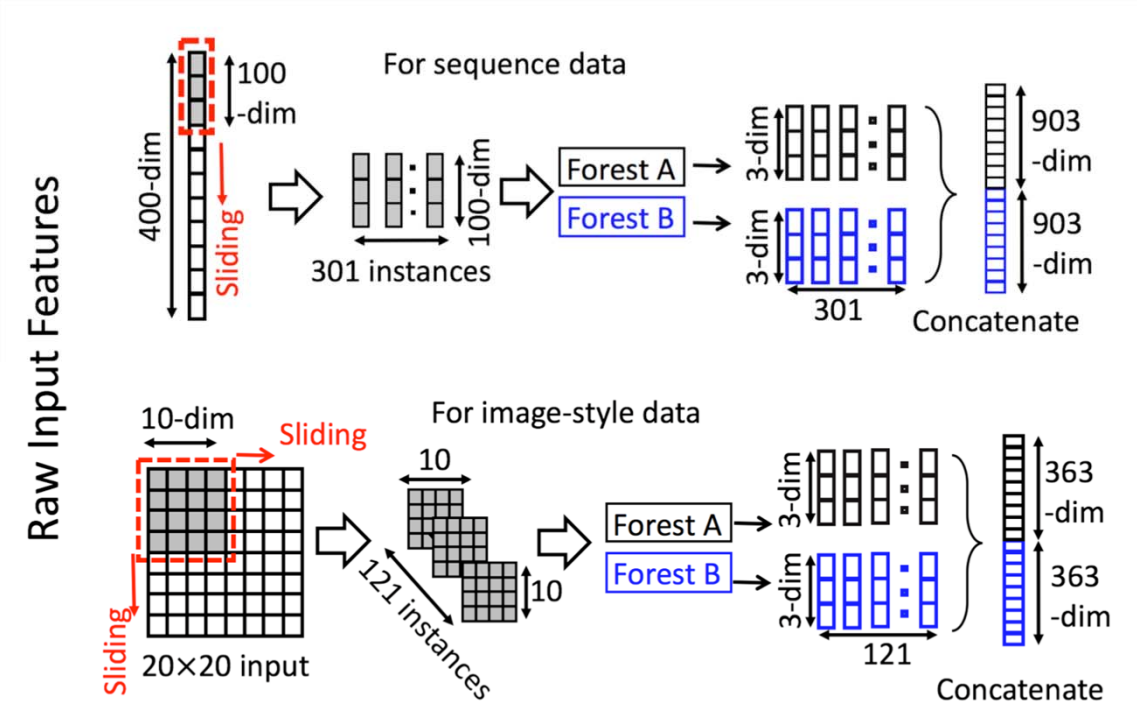
Stacking

- 级联森林结构
  - 每一层包含两类不同的随机森林结构（蓝色和黑色），形成互补



Stacking

- 多粒度扫描
  - 为了更好处理级联森林中concat操作产生的高维度特征，提出了多粒度扫描方法，减小每个随机森林网络处理的特征维度

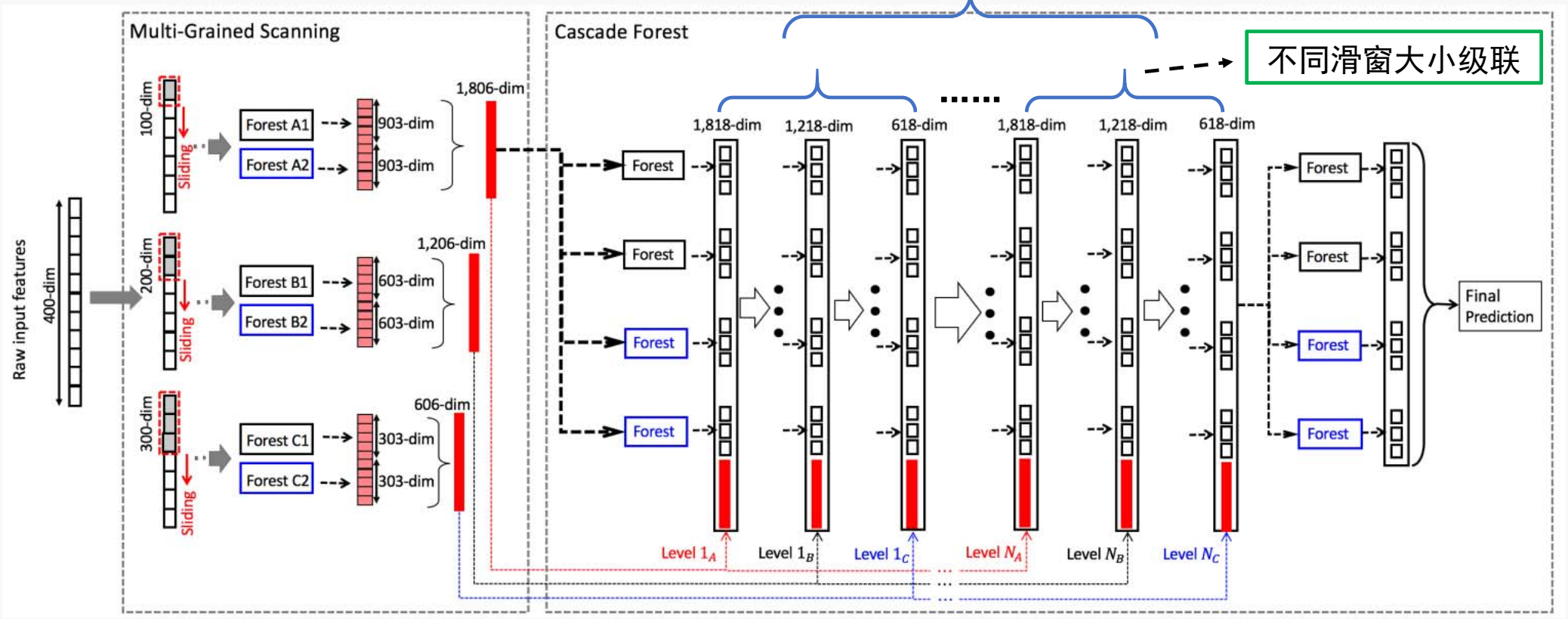


Stacking

Multi-Grained Cascade Forest (gcForest) 整体结构

不同滑窗大小的多组随机森林提取特征

多组不同滑窗大小级联



思考题

- 分析集成学习的训练开销
  - 使用单一学习器时通常需使用交叉验证等方法来确定参数值，这事实上已使用了不同参数训练出多个学习器，只不过最终仅选择其中一个学习器进行使用，而集成学习则相当于把这些学习器都利用起来。由此可看出，集成学习技术的实际训练开销并不比使用单一学习器大很多
- 是否可以通过分布式计算加速集成学习的训练速度
  - Bagging：各个基学习器训练是独立的，可以
  - Boosting：当前学习器需基于上一个学习器，不可以
  - Stacking：同一层次的学习器之间如果训练是独立的，则可以；但只有上一层次的训练完成了才能进行下一层次的训练，因此层与层之间不可以

思考题

- 利用集成学习的思想，如何对支持向量机进行集成学习？