第十四章：集成学习

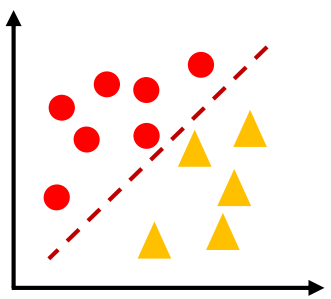
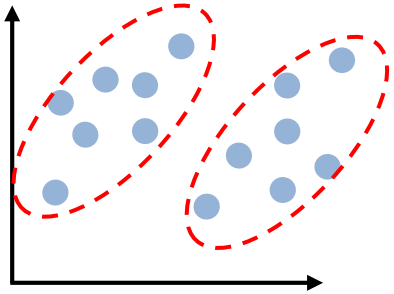
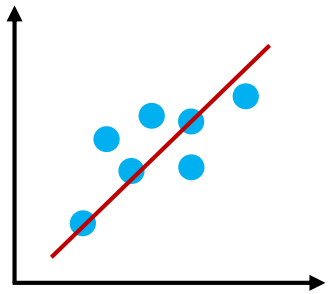
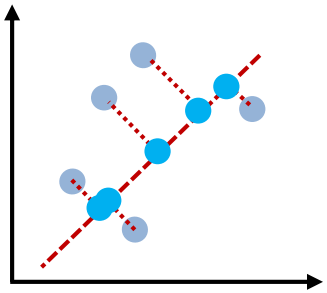
Chapter 14: Ensemble Learning

张永飞

2025年6月6日

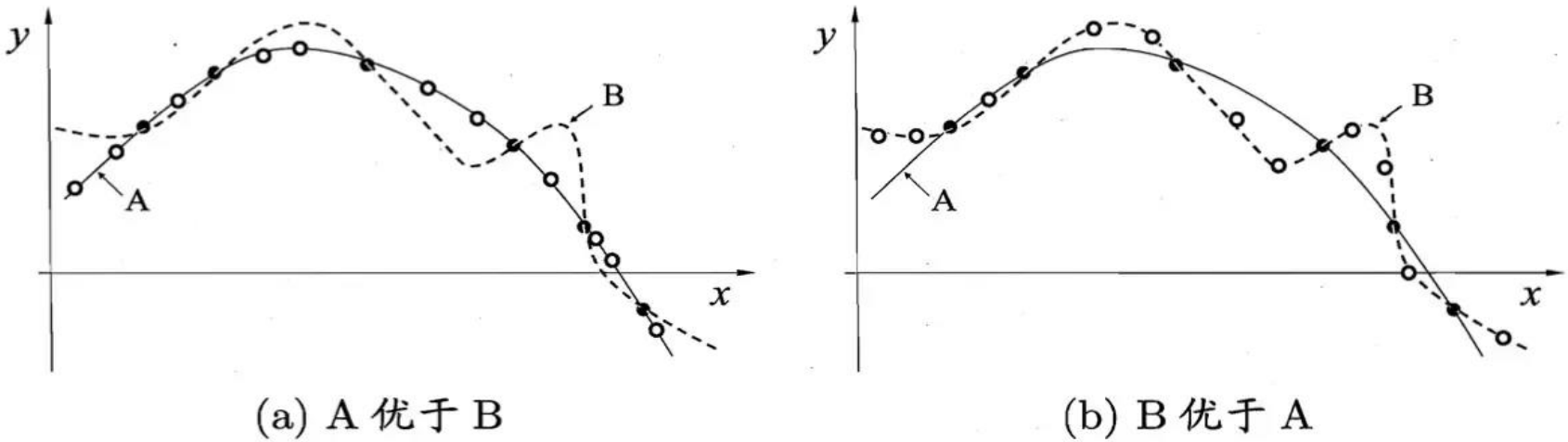
机器学习的主要研究问题

学习方式

输出变量的类型	监督学习	无监督学习
离散	分类 	聚类 
连续	回归 	降维 

传统学习面临的问题

- “没有免费的午餐 (No Free Lunch）”定理：
  没有任何情况下都最好的机器学习算法



(a) A 优于 B　　(b) B 优于 A

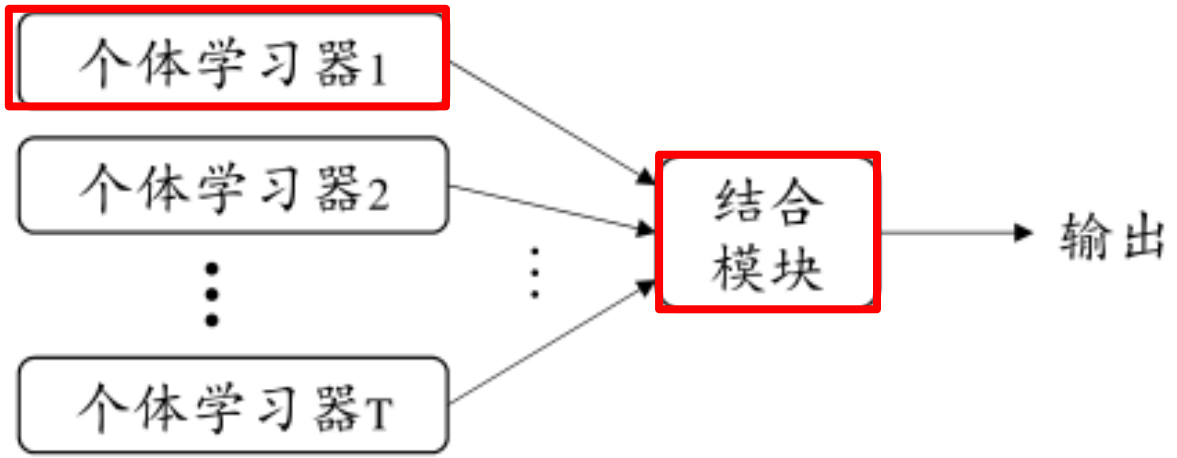
图 1　没有免费的午餐. (黑点: 训练样本; 白点: 测试样本)

“天生我材必有用！”　“三人行必有我师焉！”

什么是集成学习?

- 集成学习(Ensemble Learning)
  - 通过构建并结合多个学习器完成学习任务
  - 也称为多分类器系统(Multi-Classifier System)、基于委员会的学习(Committee based Learning)等

同质或异质



什么是集成学习?

- 通过将多个学习器进行集成，常可获得比单一学习器显著优越的泛化性能，这对弱学习器尤为明显
  - **弱学习器**：准确率仅比随机猜测略高的学习器
    其训练通常更简单，容易得到
  - **强学习器**：准确率高并能在多项式时间内完成的学习器
    其训练往往非常复杂，不容易得到

弱学习器 —集成→ 强学习器

强学习器 —集成→ 更强的学习器

集成学习基本概念

三个臭皮匠（裨将），

顶个/赛过诸葛亮!

类似谚语：“众人拾柴火焰高”、“人多出韩信”

集成学习基本概念

- 曹操：三发《求贤令》、《求言令》，唯才是举，终成霸业

集成学习基本概念

- 现代科学和工程问题：复杂，需多学科交叉融合、多方面人才协作
  - 例：“两弹一星”精神：“…，大力协同、勇于登攀”

“两弹一星”

“两弹一星”元勋

个体与集成

- 多个学习器一定比单一学习器性能好吗？ NO
  - 简单示例：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所示，其中√表示分类正确，×号表示分类错误，集成结果通过投票(Voting)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

  - 如何获得比最好的单一学习器更好的性能？

集成个体：好而不同

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，可知集成分类器的错误率为：

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

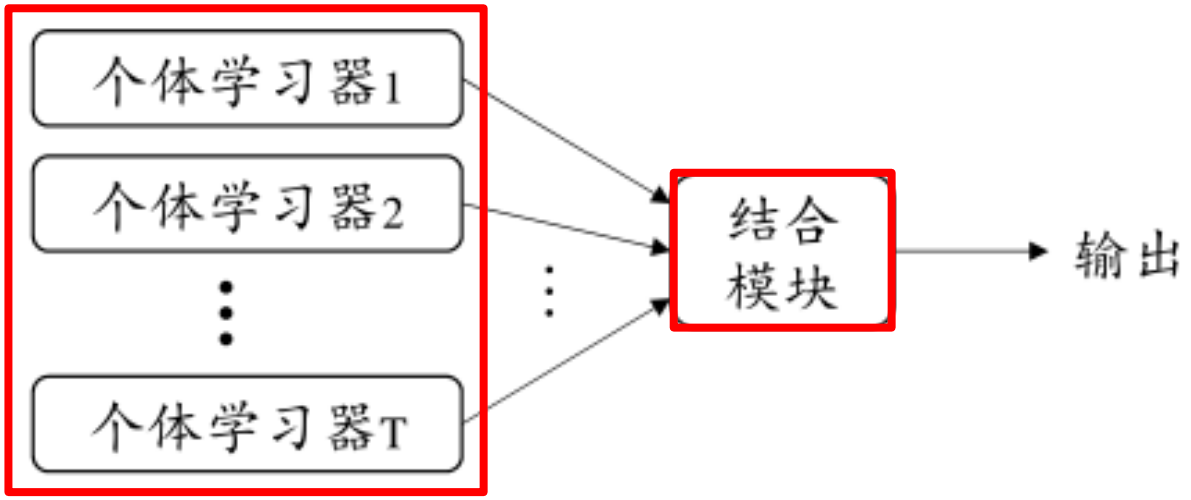
  - 则由Hoeffding不等式可知：

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

  - 可见，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)



集成学习的研究核心：

如何产生“好而不同”的个体学习器?

如何组合多个个体学习器形成集成学习器?

多样性

- **多样性度量(Diversity Measure)**
  - 用于度量集成中个体学习器的多样性
  - 考虑个体学习器的两两相似/不相似性

给定数据集$D = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}$，对二分类任务$y_i \in \{-1, +1\}$

分类器h_i与h_j的预测结果联立表(Contingency Table)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

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

- **多样性度量(Diversity Measure)**
  - **不合度量(Disagreement Measure)** $dis_{ij}=\frac{b+c}{m}$

    值域[0, 1]，值越大多样性越大.
  - **相关系数(Correlation Coefficient)**

    值域[-1, 1]，学习器无关值为0；正相关值为正，否则为负.

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

进行了归一化
值越小，多样性越大

  - **Q-统计量(Q-Statistic)** $Q_{ij}=\frac{ad-bc}{ad+bc}$

    Q_{ij}**与相关系数** ρ_{ij} **符号相同，且** $|Q_{ij}|\leq|\rho_{ij}|$ **.**

多样性

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

– ***k*-统计量(Kappa-Statistic)** $\kappa=\frac{p_1-p_2}{1-p_2}$

p_1：h_ih_j一致概率 $p_1=\frac{a+d}{m}$, 两个分类器结果一致的概率

$$p(h_i=h_j)=p(h_i=1,h_j=1)+p(h_i=-1,h_j=-1)=\frac{a}{m}+\frac{d}{m}=p_1$$

p_2：h_ih_j偶然一致概率(h_ih_j独立) $p_2=\frac{(a+b)(a+c)+(c+d)(b+d)}{m^2}$

$$p(h_i=1)=\frac{a+b}{m},\quad p(h_i=-1)=\frac{c+d}{m}\qquad p(h_j=1)=\frac{a+c}{m},\quad p(h_j=-1)=\frac{b+d}{m}$$

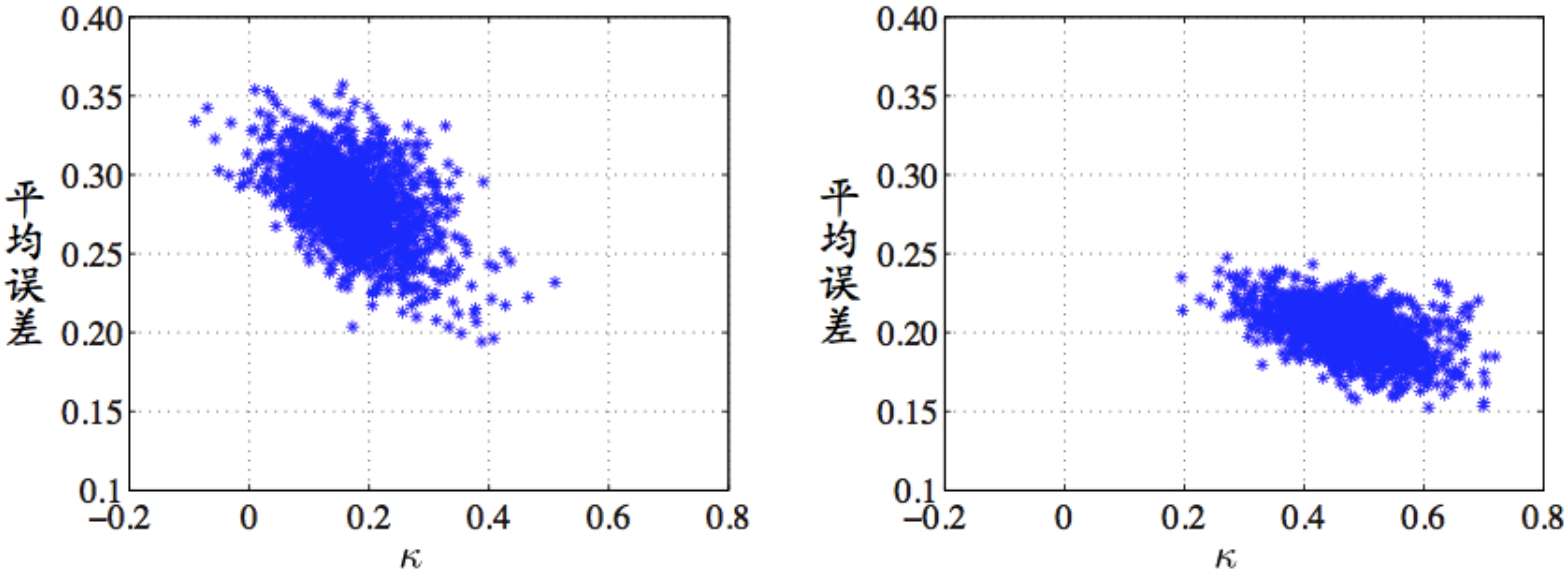
$$\hat{p}(h_i=h_j)=\hat{p}(h_i=1,h_j=1)+\hat{p}(h_i=-1,h_j=-1)$$
$$=p(h_i=1)p(h_j=1)+p(h_i=-1)p(h_j=-1)=p_2$$

通常非负，学习器完全一致时值为1；偶然一致时值为0，一致概率低于偶然时取负值.

多样性

- **多样性度量(Diversity Measure)**

  ***k*-误差图**



(a) AdaBoost 集成　　(b) Bagging 集成

横坐标是学习器的*k*值，纵坐标是其平均误差。数据点云位置越高，个体学习器准确性越低；点云位置越靠右，个体学习器多样性越小.

多样性

- 多样性增强：在学习过程引入随机性
  - 数据样本扰动
  - 输入属性扰动
  - 输出表示扰动
  - 算法参数扰动
  - 不同的多样性增强机制也可一起使用
    - Adaboost：加入了数据样本扰动
    - 随机森林：同时加入了数据样本扰动和输入属性扰动

多样性

- 数据样本扰动：通常是基于采样法
  - Bagging中的自助采样
  - Adaboost中的序列采样
  - 对数据样本扰动敏感的基学习器(不稳定基学习器)
    - 决策树，神经网络等

    效果明显
  - 对数据样本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

    效果不明显

多样性

- **输入属性扰动**：不同子空间提供观察数据的不同视角
  - 典型算法：**随机子空间算法**[Ho, 1998]

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

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

---

d'小于初始属性数d
$\mathcal{F}_t$ 包含d'个随机选取的属性，D_t仅保留 $\mathcal{F}_t$ 中的属性

对包含大量冗余属性数据，可产生多样性大的个体学习器，还因属性数减少会大幅节省时间开销；若数据只含少量属性或冗余属性较少，则不宜使用.

多样性

- **输出表示扰动**：操纵输出表示
  - 翻转法(Flipping Output)：对训练样本的类标记稍作变动【2000年Brieman提出】
    - 随机改变一些训练样本的标记（噪声，增强模型泛化性）
  - 输出调制法(Output Smearing)：对输出表示进行转化【2000年Brieman提出】
    - 将分类输出转为回归输出构建个体学习器
  - ECOC法：将原任务拆解为多个可同时求解的子任务【1995年Dietterich和Bakiri提出】
    - 利用纠错输出码将多分类任务拆解为一系列二分类任务训练基学习器

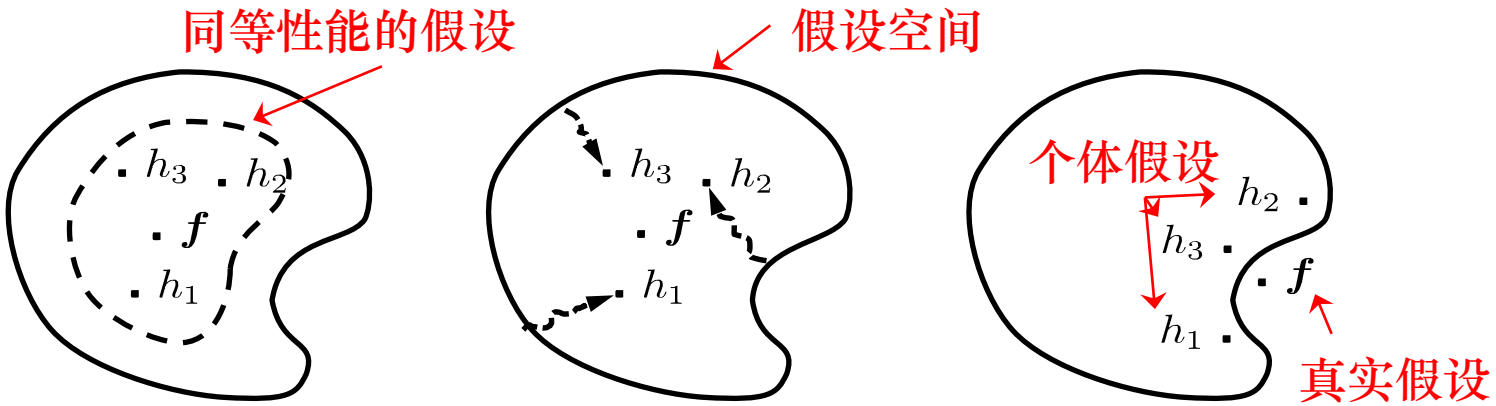
多样性

- **算法参数扰动**：随机设置不同的参数或环节
  - 参数较多时
    - 负相关法(Negative Correlation)：【1999年Liu和Yao提出】
    - 显式地通过正则化项强制个体神经网络使用不同参数
  - 参数较少时
    - 例如，可将决策树使用的属性选择机制用其他类似方式代替，信息增益、信息增益率、基尼指数等

单一学习器利用交叉验证对参数寻优，事实上相当于使用了不同参数训练学习器，最后仅选择了一个；而集成学习相当于把所有学习器都利用起来

结合策略

- 学习器的组合有三个方面的好处
  - 统计方面：减小误选假设空间导致泛化性能不佳的几率
  - 计算方面：降低陷入坏局部极小点影响泛化性能的风险
  - 表示方面：扩大假设空间(域)学习对于真实空间更好近似



(a) 统计的原因　　(b) 计算的原因　　(c) 表示的原因

h1,h2,h3 是学习结果， f 是集成结果

结合策略—平均法

- 平均法(Averaging)是数值型输出最常见的结合策略
  - 简单平均法(Simple Averaging)　个体学习器性能相近时适用

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

  - 加权平均法(Weighted Averaging) 个体学习器性能迥异时适用

【1993年Perrone和Cooper正式将其用于集成学习】

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

加权平均法是集成学习的基本出发点，各种结合方法都可视为其特例或变体，不同的集成学习方法是通过不同的方式确定加权平均法中基学习器的权重

结合策略—投票法

硬投票（类标签）和软投票（类概率）

- 投票法(Voting)是标签型输出最常见的结合策略

  标记集合$\{c_1, c_2, ..., c_N\}$，h_i在样本x上的预测$\{h_i^1(x), h_i^2(x), ..., h_i^N(x)\}$

  - 绝对多数投票法(Majority Voting)：得票超半数

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

  - 相对多数投票法(Plurality Voting)：得票最多

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

  - 加权投票法(Weighted Voting)：加权后得票最多

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

结合策略—学习法

- 当训练数据很多时采用另一个学习器进行结合

  初级学习器 vs. 次级学习器或元学习器(Meta-learner)

  – Stacking是学习法的典型代表【1992年Wolpert提出】

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:

从初始数据集训练初级学习器

```
for t = 1, ..., T:      % Train a first-level learner by applying the
    h_t = L_t(D);       % first-level learning algorithm L_t
end
```

生成次级数据集：初级学习器的输出被当作样例输入特征，继承初始样本标记.

```
D' = ∅;                 % Generate a new data set
for i = 1, ..., m:
    for t = 1, ..., T:
        z_it = h_t(x_i);
    end
    D' = D' ∪ ((z_i1, ..., z_iT), y_i);
end
```

从次级数据集训练次级学习器

```
h' = L(D');             % Train the second-level learner h' by applying
                        % the second-level learning algorithm L to the
                        % new data set D'.
```

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

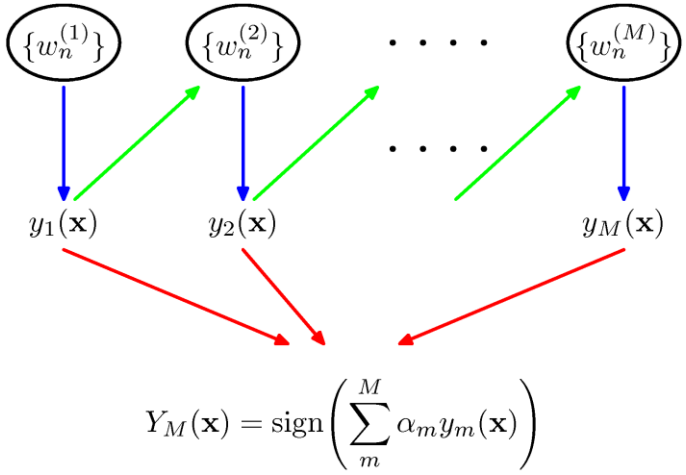
集成学习方法

- 根据个体学习器生成方式不同，形成两大类方法
  - 串行化方法：个体学习器间存在强依赖关系

    一轮一轮再一轮
    - 典型算法：提升Boosting算法(Adaboost)
  - 并行化方法：个体学习器间不存在强依赖关系
    - 典型算法：装袋Bagging算法(随机森林Random Forest)

串行化方法 Boosting

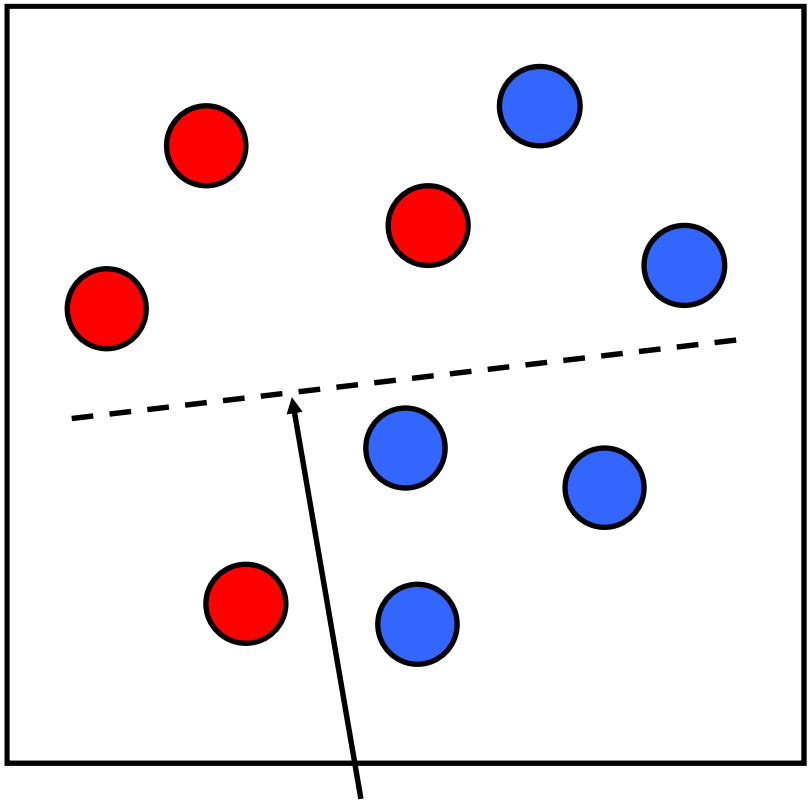
- 一族可将弱学习器提升为强学习器的算法
  - 串行生成
  - 个体学习器存在强依赖关系
  - 每次调整训练数据的样本分布

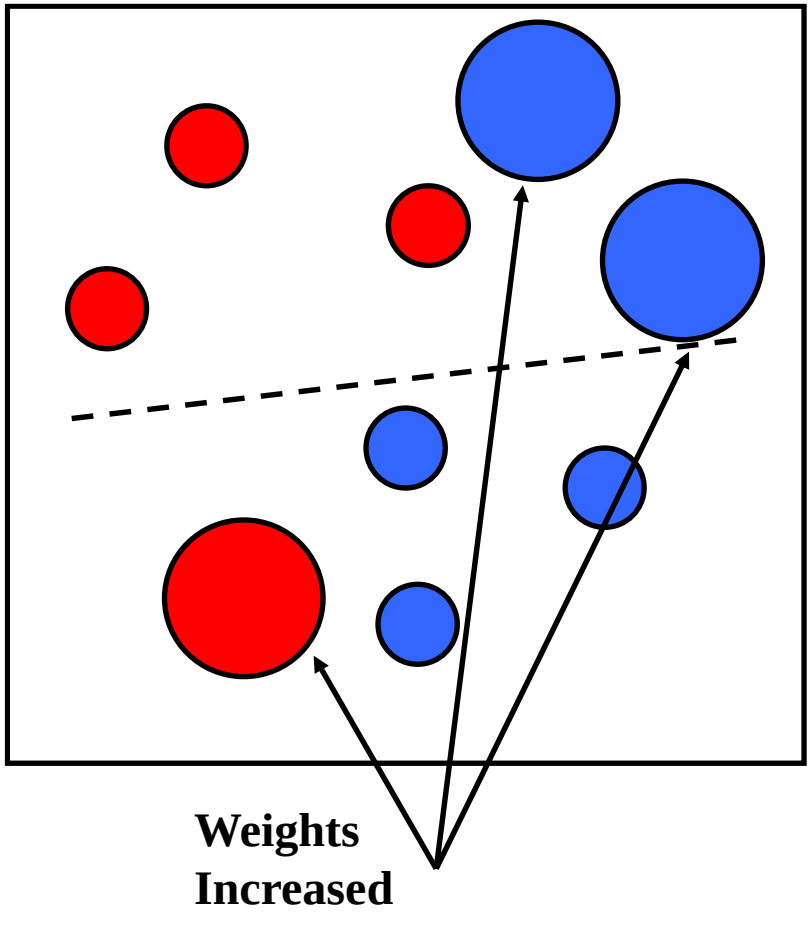


- **基本思想**：

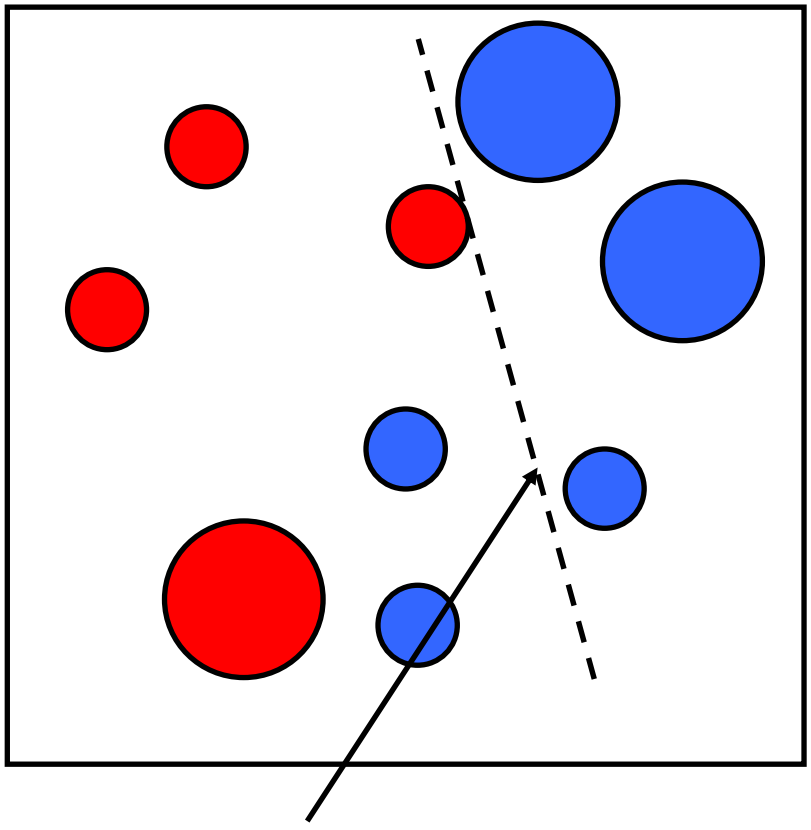
先从初始数据集训练一个基学习器，再根据其对训练样本分布（权重）进行调整，使先前错分样本在后续受到更多关注，然后基于调整后的样本分布训练下一个基学习器；重复进行直至基学习器数目达到预先指定值；最终将这些基学习器加权结合

Boosting-基本思想示意

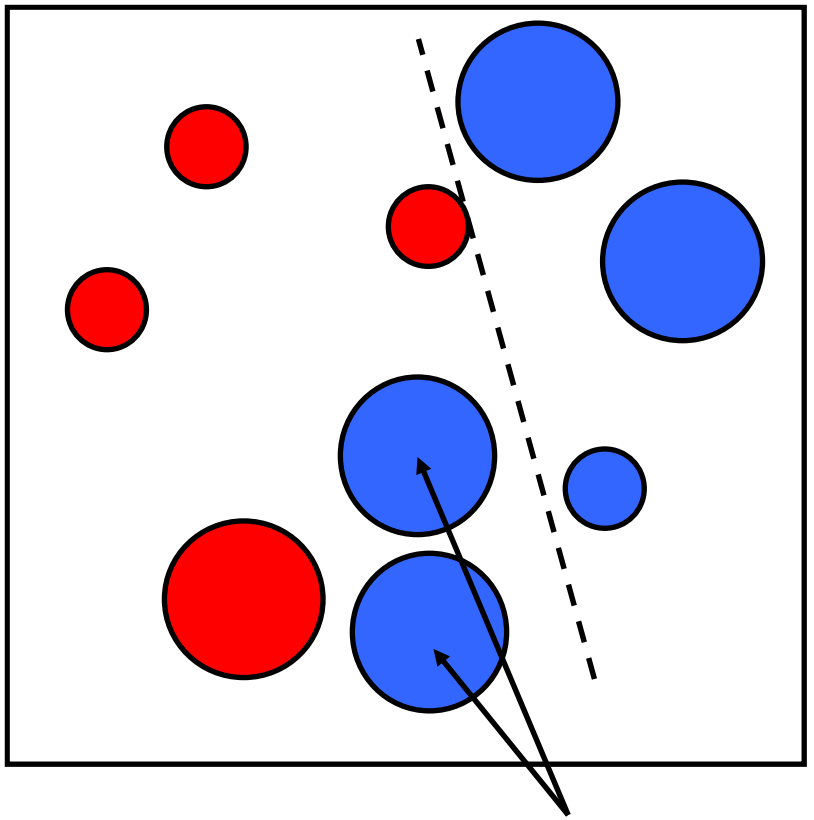




Boosting-基本思想示意

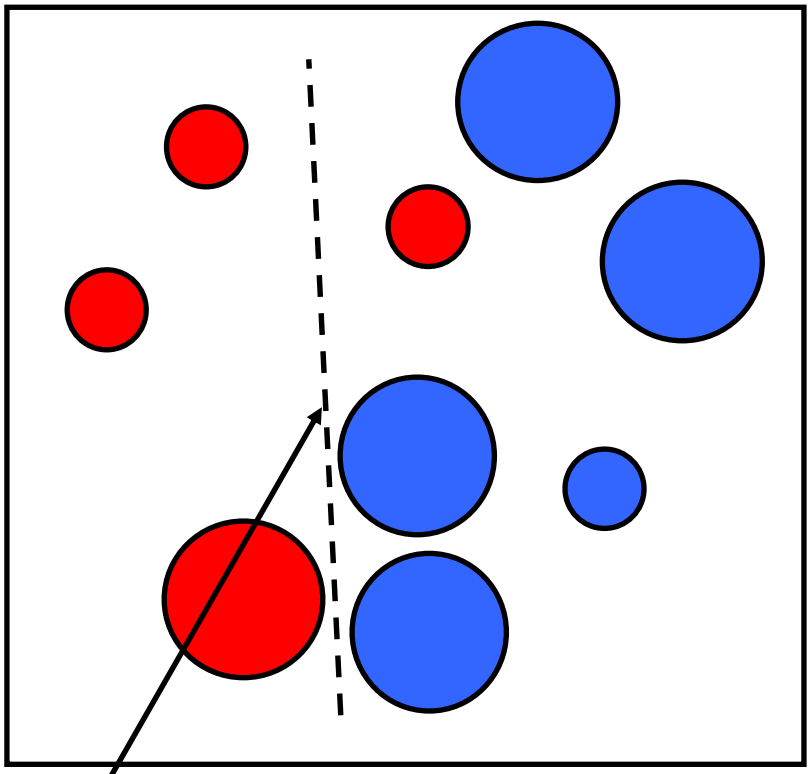


Weak Classifier 2

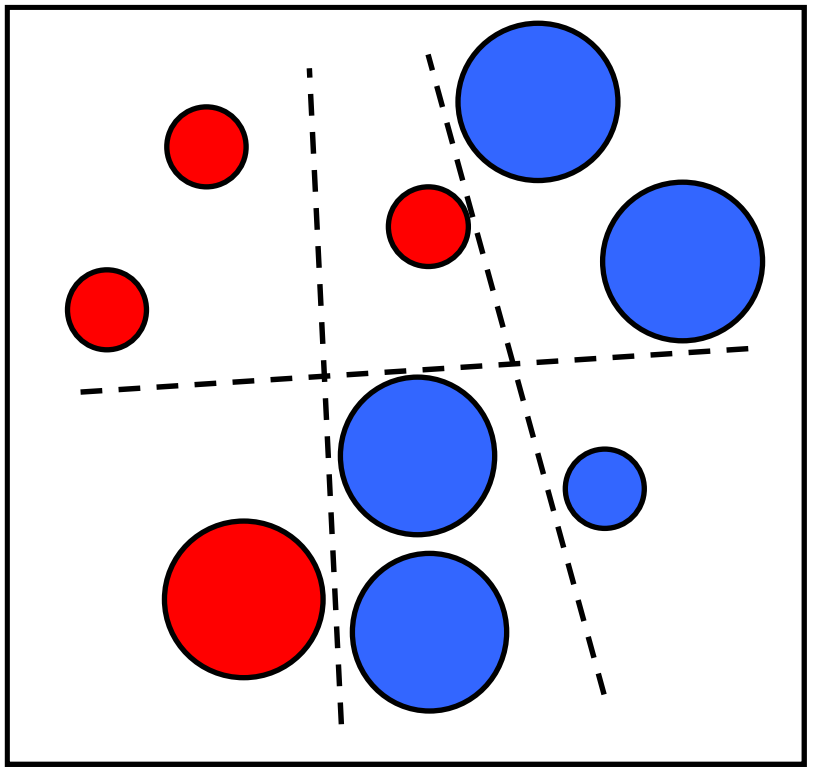


Weights Increased

Boosting-基本思想示意



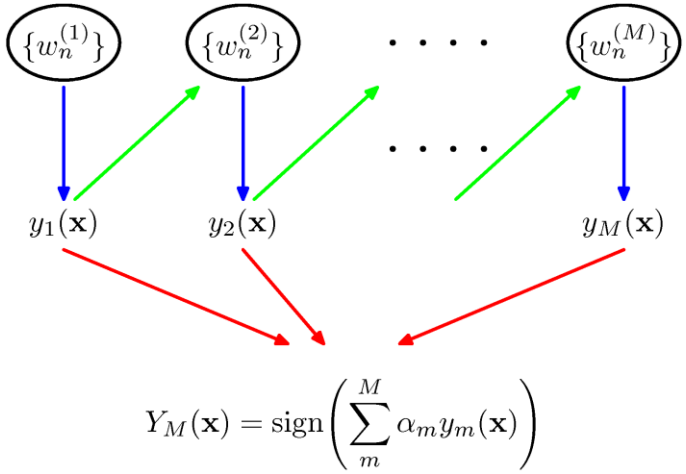
Weak
Classifier 3



Final classifier is
a combination of weak
classifiers

串行化方法 Boosting

- 一族可将弱学习器提升为强学习器的算法
  - 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



- **基本思想**：

先从初始数据集训练一个基学习器，再根据其对训练样本分布（权重）进行调整，使先前错分样本在后续受到更多关注，然后基于调整后的样本分布训练下一个基学习器；重复进行直至基学习器数目达到预先指定值；最终将这些基学习器加权结合

Boosting算法

- **数据分布调整：基学习器学习特定的数据分布**
  - **重赋权法(Re-weighting)**

  在每轮根据样本分布为每个训练样本重新赋予权重
  - **重采样法(Re-sampling)**

  在每轮根据样本分布对训练集重新采样形成新的训练集

  把错的题多做几次（多选几次）

  注意：Boosting每轮检查当前生成的基学习器是否满足优于随机猜测的基本条件，若不满足，此基学习器被抛弃，学习过程停止。

Adaboost算法

- **Boosting族算法最著名的代表** 【1997年Freund和Schapire提出】

f：真实函数

$y \in \{-1, +1\}$

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: $\mathcal{D}_1(\boldsymbol{x}) = 1/m.$

初始化样本权值分布

2: **for** $t = 1, 2, \ldots, T$ **do**

3: $h_t = \mathfrak{L}(D, \mathcal{D}_t)$;

基于分布D_t训练学习器h_t

4: $\epsilon_t = P_{\boldsymbol{x} \sim \mathcal{D}_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$;

估计h_t误差

5: **if** $\epsilon_t > 0.5$ **then break**

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

确定学习器h_t的权重

误差越小，权重越大；误差越大，权重越接近0

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

更新样本分布
(Z_t规范化因子)

8: **end for**

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

推导详见P173-176

Boosting算法

- 特点总结：
  - 基本思想是用贪心法最小化损失函数，
  - 主要关注降低偏差：顺序串行地最小化损失函数，基于弱学习器逐步构造出很强的集成学习器，bias自然逐步下降
  - 但是由于模型的相关性很强，因此不能显著降低方差
  - 所以boosting主要靠降低偏差来提升预测精度
- 基学习器特点：高偏差，低方差的弱模型

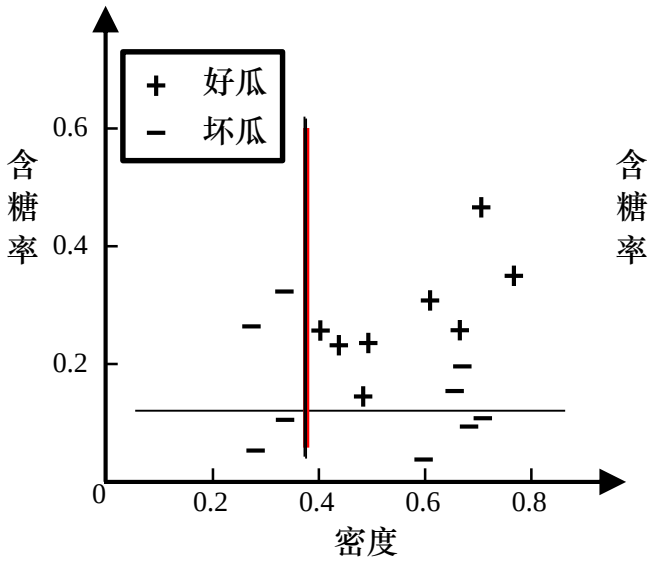
Boosting算法

- 示例-西瓜分类

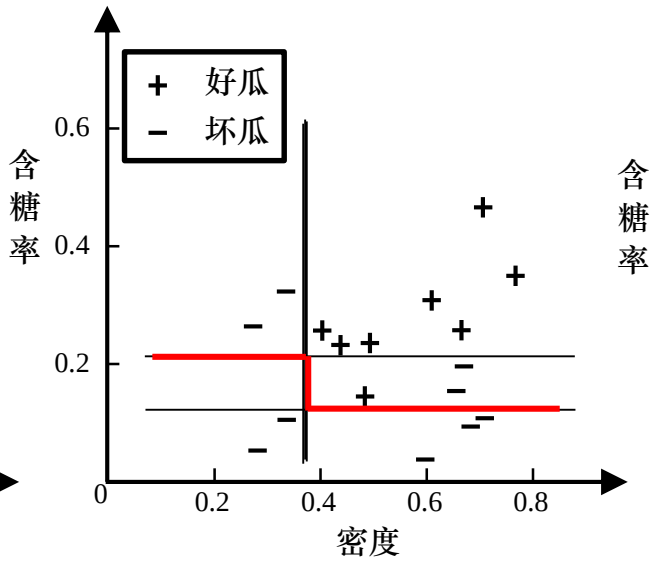
编号	色泽	根蒂	敲声	纹理	脐部	触感	密度	含糖率	好瓜
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	0.697	0.460	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	0.774	0.376	是
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	0.634	0.264	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	0.608	0.318	是
5	浅白	蜷缩	浊响	清晰	凹陷	硬滑	0.556	0.215	是
6	青绿	稍蜷	浊响	清晰	稍凹	软粘	0.403	0.237	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	0.481	0.149	是
8	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	0.437	0.211	是
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	0.666	0.091	否
10	青绿	硬挺	清脆	清晰	平坦	软粘	0.243	0.267	否
11	浅白	硬挺	清脆	模糊	平坦	硬滑	0.245	0.057	否
12	浅白	蜷缩	浊响	模糊	平坦	软粘	0.343	0.099	否
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	0.639	0.161	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	0.657	0.198	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	0.360	0.370	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	0.593	0.042	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	0.719	0.103	否

Boosting算法

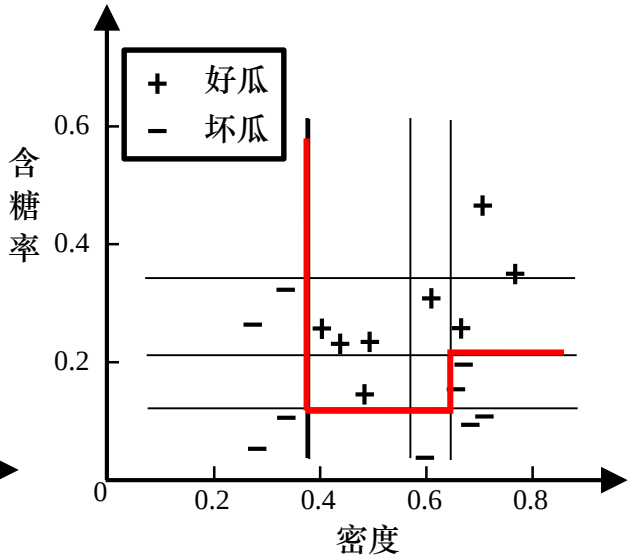
- 示例-西瓜分类



(a) 3个基学习器



(b) 5个基学习器



(c) 11个基学习器

并行化方法　Bagging算法

- **并行式集成学习最著名的代表性方法**【1996年Breiman提出】

  名字由来： 由Bootstrap AGGregatING缩写而来

  基本思想：

  利用自助法采样(Bootstrap Sampling)可构造T个含m个训练样本的采样集，基于每个采样集训练出一个基学习器，再将它们进行结合

  在对预测输出结合时，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算法

- 并行式集成学习最著名的代表性方法【1996年Breiman提出】
  - 自助法采样 (Bootstrap Sampling)

给定包含m个样本的数据集D，对其进行采样产生数据集D'：

每次随机从D中挑选一个样本，将其拷贝至D'，这个过程重复执行m次后，就得到了包含m个样本的数据集D'。显然，D中有一部分样本会在D'中多次出现，而有一部分样本则不会出现。

样本在m次采样中始终不被采到的概率是 $(1-\frac{1}{m})^m$, 其极限:

将D'用做训练集; $D \backslash D'$用作测试集

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

产生不同训练集对集成学习有好处，但改变数据分布会引入偏差。

【1993年Efron和Tibshirani提出】

Bagging算法

- **并行式集成学习最著名的代表性方法【1996年Breiman提出】**

算法流程：

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.

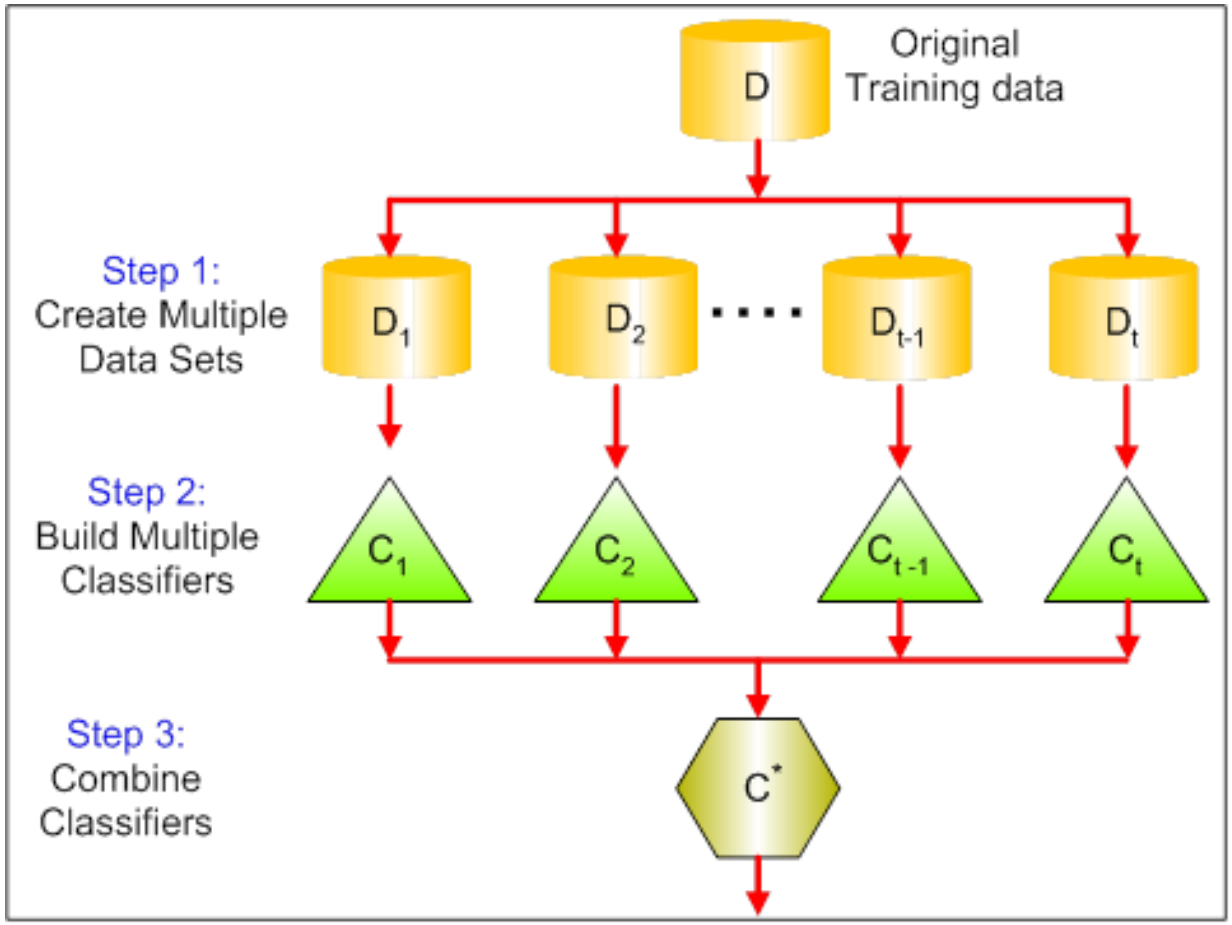
过程：

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

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

Bagging算法

- 并行式集成学习最著名的代表性方法【1996年Breiman提出】



Bagging算法

- **算法特点**
  - **时间复杂度低**：集成与直接训练一个学习器复杂度同阶

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$T(O(m)+O(s))$；

  - 可以直接用于多分类、回归等任务；
  - 可包外估计(Out-of-Bag Estimate)泛化性能.

$H^{oob}(x)$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：

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

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算法

- 特点总结
  - 主要关注降低方差，即通过多次重复训练提高稳定性，在易受样本扰动的学习器上效用更为明显(如不剪枝的决策树、神经网络等)
  - 在Bagging中，每个模型的偏差方差近似相同，但是互相相关性不太高，因此一般不能降低偏差
  - 对噪声数据相对鲁棒
- 基学习器特点：低偏差、高方差(通常用强学习器)

并行化方法 随机森林算法

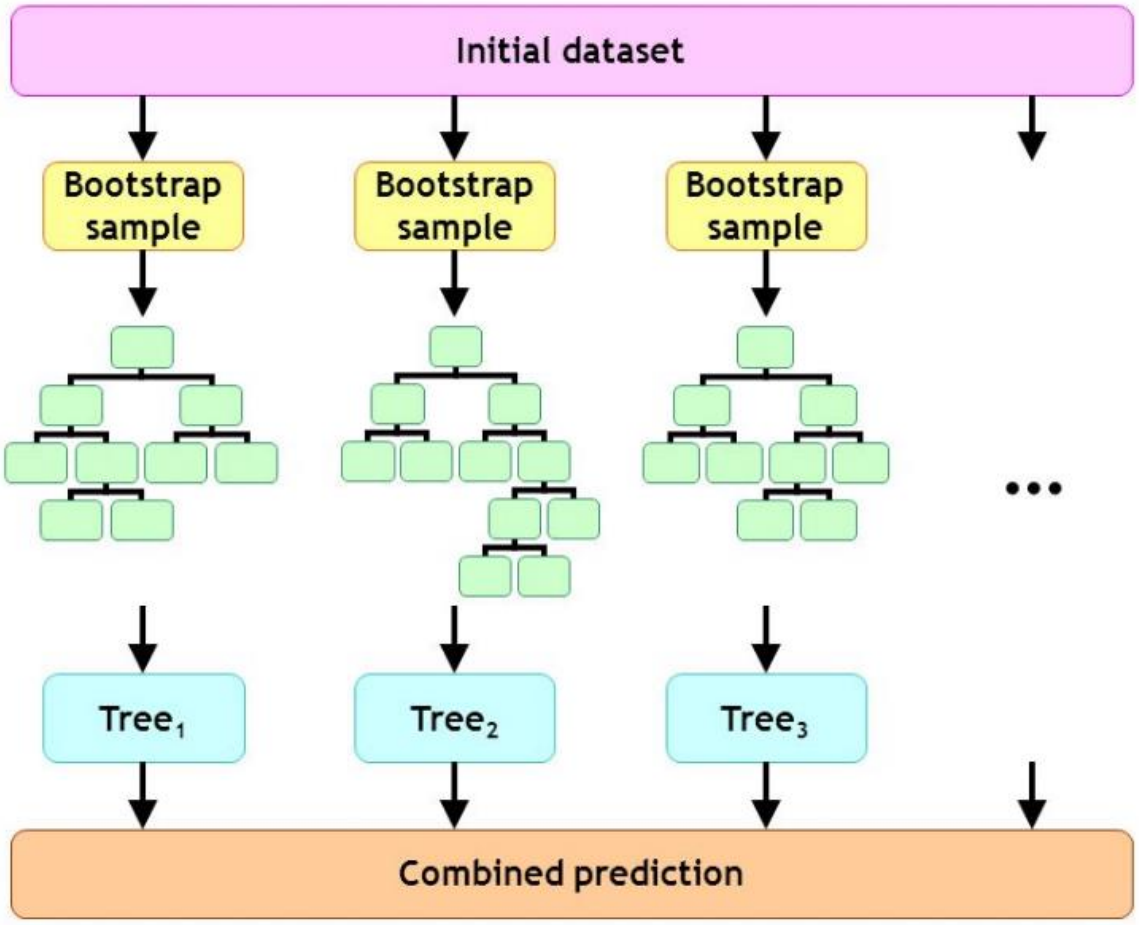
- Bagging方法的一种扩展变体 【2001年Breiman提出】
  - Random Forest，简称RF
  - 以决策树为基学习器
- **基本思想**:
  - **数据集的随机选择**：自助采样法
  - **待选属性的随机选择**：对基决策树的每个结点，先从该结点的(d个)属性集合中随机选择一个包含 k 个属性的子集，再从这个子集选择一个最优属性用于划分，一般情况下推荐 $k=\log_2 d$

并行化方法 随机森林算法

- **算法流程**：
  - 从原始数据集中每次随机有放回抽样选取与原始数据集相同数量的样本数据，构造数据子集；
  - 每个数据子集从所有待选择的特征中随机选取一定数量的最优特征作为决策树的输入特征；
  - 根据每个子集分别得到每棵决策树，由多棵决策树共同组成随机森林；
  - 最后如果是分类问题，则按照投票的方式选取票数最多的类作为结果返回；如果是回归问题，则按照平均法选取所有决策树预测的平均值作为结果返回

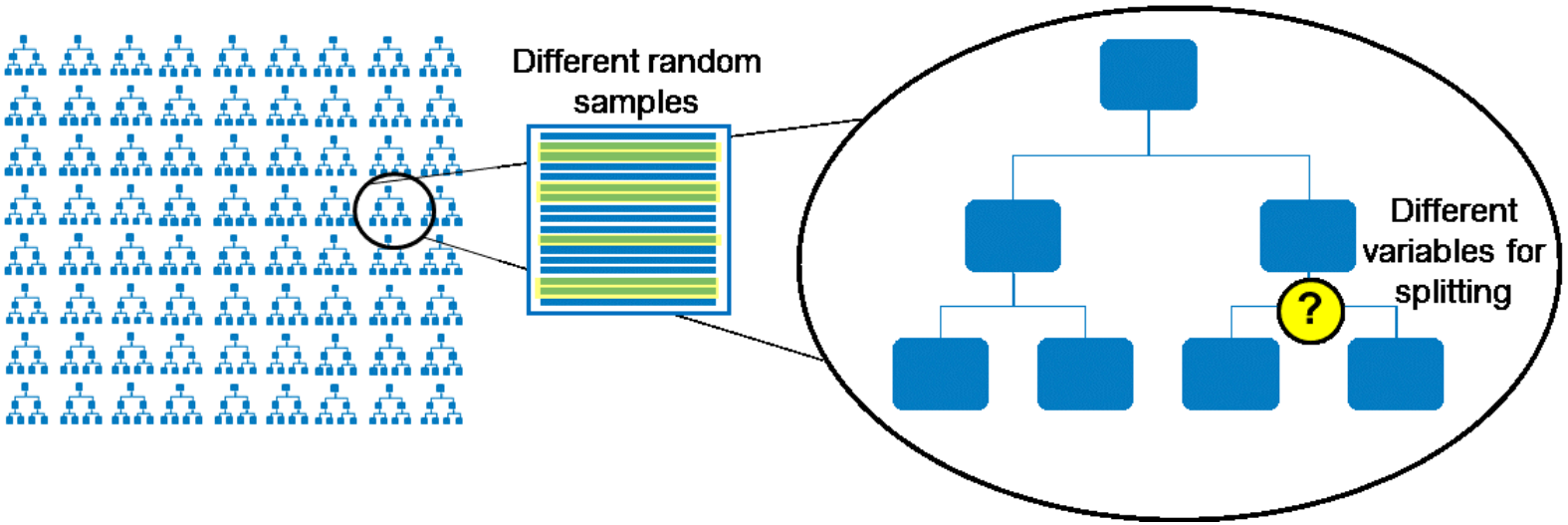
随机森林算法

- **Bagging方法的一种扩展变体** 【2001年Breiman提出】



随机森林算法

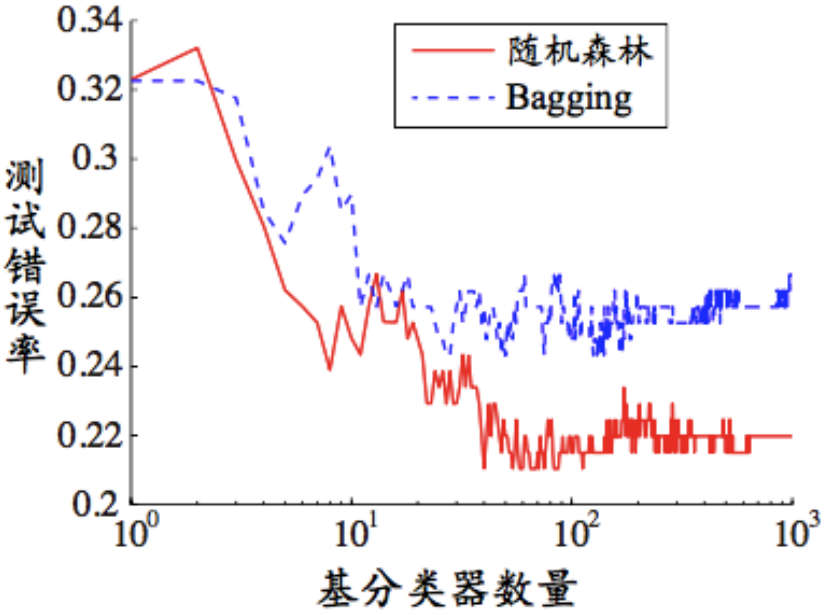
- 算法特点
  - 基学习器多样性通过样本扰动和属性扰动实现
  - 算法简单、容易实现、计算开销小
  - 性能强大，被誉为“代表集成学习技术水平的方法”



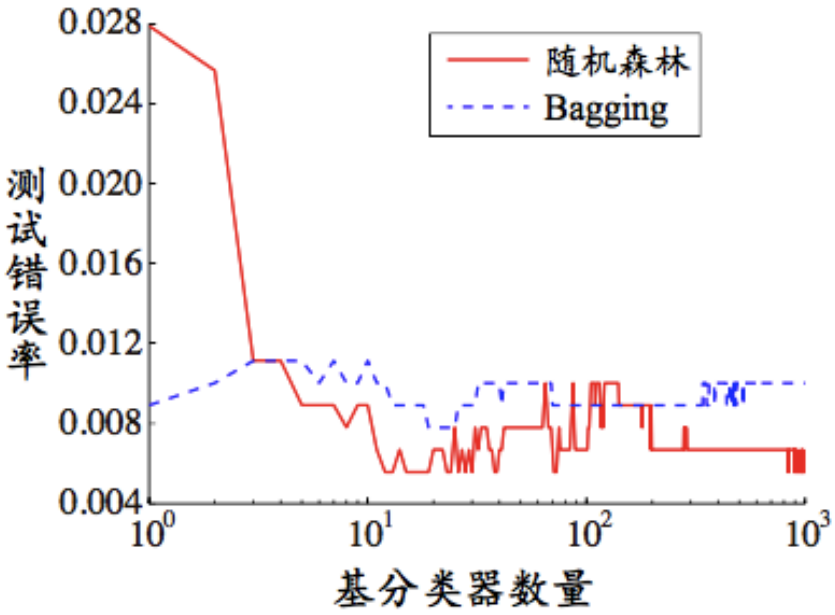
Bagging vs. RF

- RF与Bagging收敛性相似

UCI数据



(a) glass 数据集



(b) auto-mpg 数据集

- RF训练效率优于Bagging (随机型 vs. 确定型)

本节课程结束!

下节课再见!