华中科技大学
电子信息与通信学院
School of Electronic Information and Communications

机器学习导论

– Introduction to Machine Learning

第 8 章：集成学习

(Chapter 8: Ensemble Learning)

华中科技大学电信学院

王邦　博士, 教授博导

wangbang@hust.edu.cn

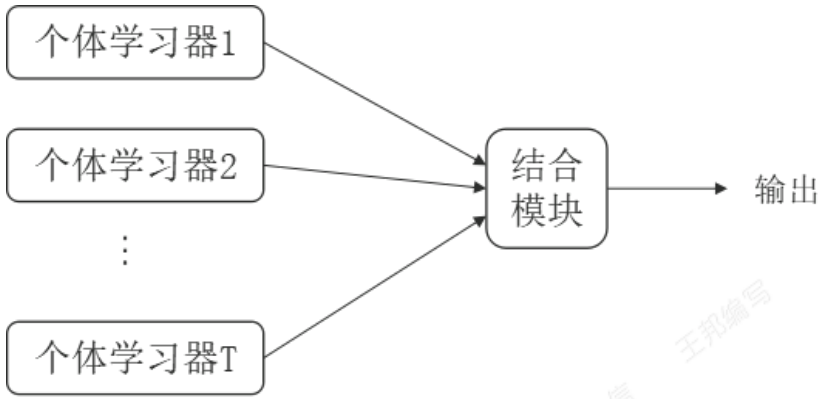
第 08 章：集成学习

个体与集成 (1)

- **集成学习** (ensemble learning)：通过构建并结合多个学习器来完成学习任务，常可获得比单一学习器显著优越的泛化性能，有时也被称为多分类器系统 (multi-classifier system) 等。
- **集成学习的一般结构**：先产生一组“个体学习器”(individual learner)，再用某种策略将它们结合起来。个体学习器通常由一个现成的学习算法从训练数据产生。

  个体学习器可以是同质的 (homogeneous)，即每个个体学习器采用相同的学习算法，称之为基学习器 (base leaner)。

  个体学习器也可以是异质的 (heterogeneous)，即个体学习器可以采用不同的学习算法，此时每个个体学习器称之为组件学习器 (component leaner)。



集成学习示意图

个体与集成 (2)

例子：在二分类任务中，假定三个分类器在三个测试样本上的表现如下表所示，其中 $\surd$ 表示分类正确，$\times$ 表示分类错误，集成学习的结果通过投票法产生，即“少数服从多数”。

- 表 (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%；
- 表 (b) 中，三个分类器没有差别，集成之后性能没有提高；
- 表 (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。

	测试例 1	测试例 2	测试例 3
h_1	$\surd$	$\surd$	$\times$
h_2	$\times$	$\surd$	$\surd$
h_3	$\surd$	$\times$	$\surd$
集成	$\surd$	$\surd$	$\surd$

(a) 集成提升性能

	测试例 1	测试例 2	测试例 3
h_1	$\surd$	$\surd$	$\times$
h_2	$\surd$	$\surd$	$\times$
h_3	$\surd$	$\surd$	$\times$
集成	$\surd$	$\surd$	$\times$

(b) 集成不起作用

	测试例 1	测试例 2	测试例 3
h_1	$\surd$	$\times$	$\times$
h_2	$\times$	$\surd$	$\times$
h_3	$\times$	$\times$	$\surd$
集成	$\times$	$\times$	$\times$

(c) 集成起负作用

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”(diversity)，即学习器间具有差异。

个体与集成 (3)

例子分析： 考虑一个二分类问题 $y \in \{-1, +1\}$ 和真实函数 f，假设基分类器的错误率为 ϵ，且 $\epsilon < 0.5$ （即至少比随机猜测要好）。对每个基分类器 h_i，有

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

假设集成通过简单投票法结合 T 个分类器，若有超过半数的基分类器正确，则集成分类就正确：

$$H(x) = \text{sign}(\sum_{\text{i=1}}^{\text{T}} \text{h}_\text{i}(\text{x})).$$

假设抛硬币正面朝上的概率为 p，反面朝上的概率为 $1-p$。令 $H(n)$ 代表抛 n 次硬币所得正面朝上的次数，则最多 k 次正面朝上的概率为：

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

对 $\delta > 0$，$k = (p-\delta)n$，**Hoeffding 不等式**

$$P\left(H(n) \leq (p-\delta)n\right) \leq e^{-2\delta^2 n}.$$

假设基分类器的错误率相互独立，则由 **Hoeffding 不等式**，令 $(p-\delta)T = \lfloor \frac{T}{2} \rfloor$，可得集成的错误率为：

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

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 (4)

上述分析有一个关键假设：基学习器的误差相互独立。在现实任务中，个体学习器是为解决同一个问题训练出来的，不可能相互独立！

事实上，个体学习器的“准确性”和“多样性”存在冲突。一般的，准确性很高之后，要增加多样性就需牺牲准确性。

> 如何产生并结合“好而不同”的个体学习器，正是集成学习研究的核心。

两大类集成学习方法：

- 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（如*Boosting*）
- 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（如*Bagging*和“随机森林”）

Boosting 算法 (1)

Boosting是可将弱学习器（指泛化性能略优于随机猜测的学习器）提升为强学习器的一族算法。**基本工作机制如下：**

- 从初始训练集训练出一个基学习器 h_t，评估 h_t 性能，并计算 h_t 的权重 α_t；
- 基于 h_t 对训练样本分布进行调整，即区别对待在 h_t 中做对和做错的训练样本；
- 基于调整后的样本分布来训练下一个基学习器 h_{t+1}；
- 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：$H(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x))$。

Boosting 算法要求基学习器能对特定的数据分布进行学习，这可通过“**重赋权法**”(re-weighting) 实施，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，则可通过“**重采样法**”(re-sampling) 来处理，即在每一轮学习中，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。

从偏差一方差分解的角度看，Boosting 主要关住降低偏差，因此 Boosting 能基于泛化性能相当弱的学习器构建出很强的集成。

AdaBoost (1)

Boosting 族算法最著名的代表是 **AdaBoost**，其算法描述如下图所示。

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：

初始化样本权值分布.
1: $\mathcal{D}_1(\boldsymbol{x}) = 1/m$.

基于分布 $\mathcal{D}_t$ 从数据集 D 中训练出分类器 h_t.
2: **for** $t = 1, 2, \ldots, T$ **do**
3: $\quad h_t = \mathfrak{L}(D, \mathcal{D}_t)$;

估计 h_t 的误差.
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**

确定分类器 h_t 的权重.
6: $\quad \alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$;

更新样本分布，其中 Z_t 是规范化因子，以确保 $\mathcal{D}_{t+1}$ 是一个分布.
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： $H(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}) \right)$

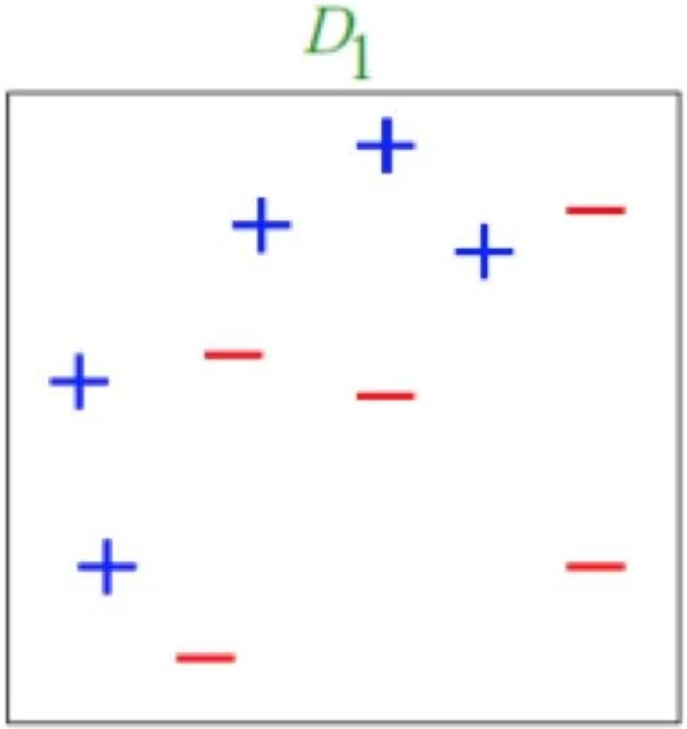
来源：周志华，《机器学习》，清华大学出版社

第一个基分类器 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 α_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 α_t 应使得 $\alpha_t h_t$ 最小化指数损失函数。

AdaBoost 图解 (1)

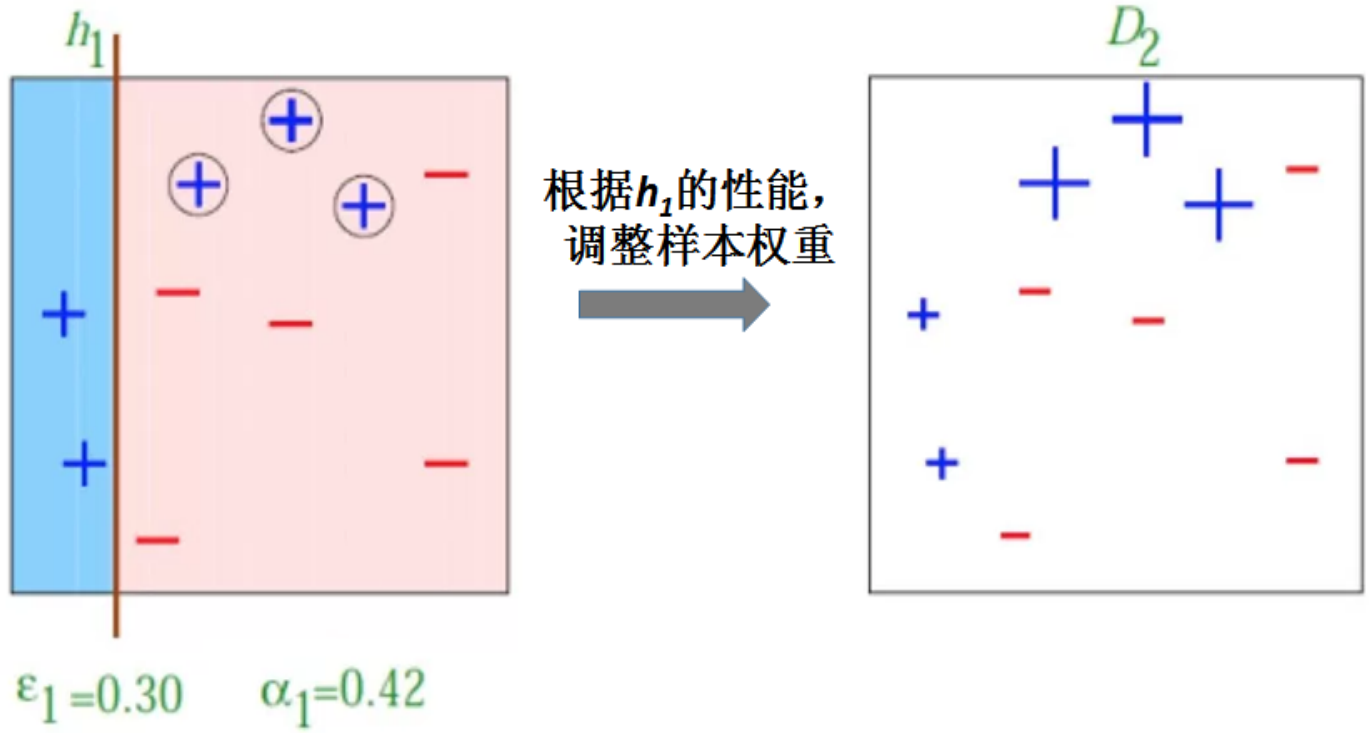
AdaBoost 图解： 举一个简单的例子来看看 AdaBoost 的实现过程。图中共 10 个训练数据，其中“+”和 “-”分别表示两种类别。基分类器使用水平或者垂直的直线来进行分类。

开始每个样本的权值设为 $1/10$，得到第一个样本分布 $\mathcal{D}_1$。图中表示为所有样本的大小相同。



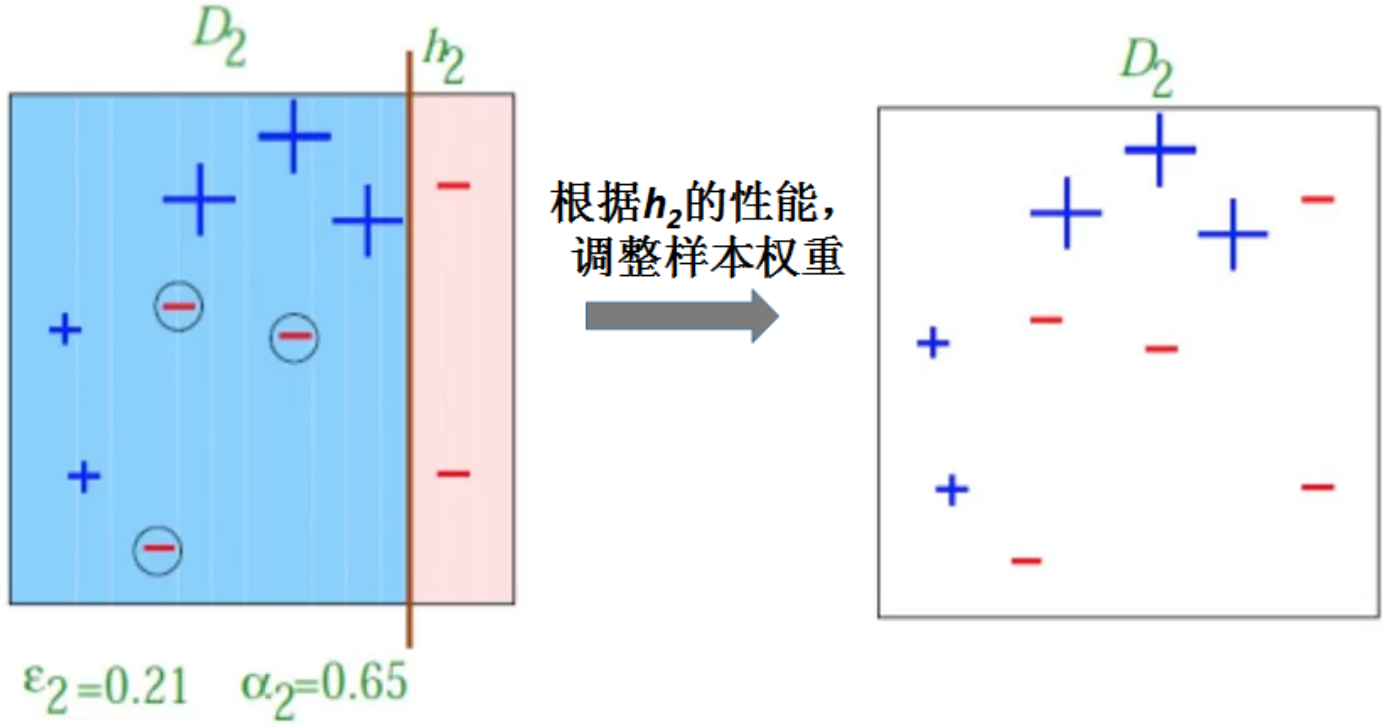
AdaBoost 图解 (2)

AdaBoost 图解： 第一个分类器 h_1 依据初始样本分布 $\mathcal{D}_1$ 进行训练，结果如左图所示，其中右边三个样本被分类错误（左图中三个画圈的样本）。计算 h_1 的误差 ϵ_1，并依此计算 h_1 的权重 α_1，且更新样本分布。右图表示依据 h_1 的分类结果对样本权重进行了调整，其中分类正确的减少权值，分类错误的增大权值，得到下一个样本分布 $\mathcal{D}_2$。图中表示为大小的改变。



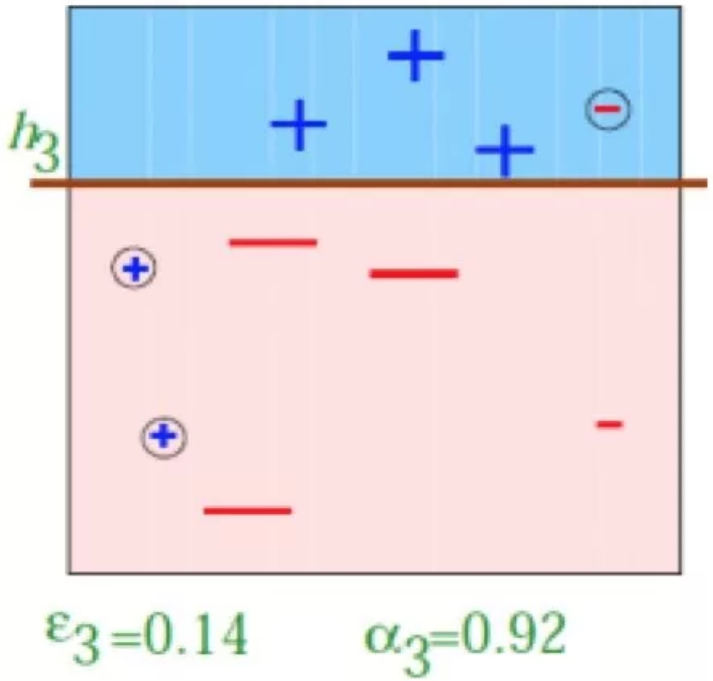
AdaBoost 图解 (3)

AdaBoost 图解： 第二个分类器 h_2 依据样本分布 $\mathcal{D}_2$ 进行训练，结果如左图所示。图中画圈的样本表示分类错误。计算 h_2 的误差 ϵ_2，并依此计算 h_2 的权重 α_2，且更新样本分布。右图表示依据 h_2 的分类结果对样本权重进行了调整，其中分类正确的减少权值，分类错误的增大权值，得到下一个样本分布 $\mathcal{D}_3$。图中表示为大小的改变。



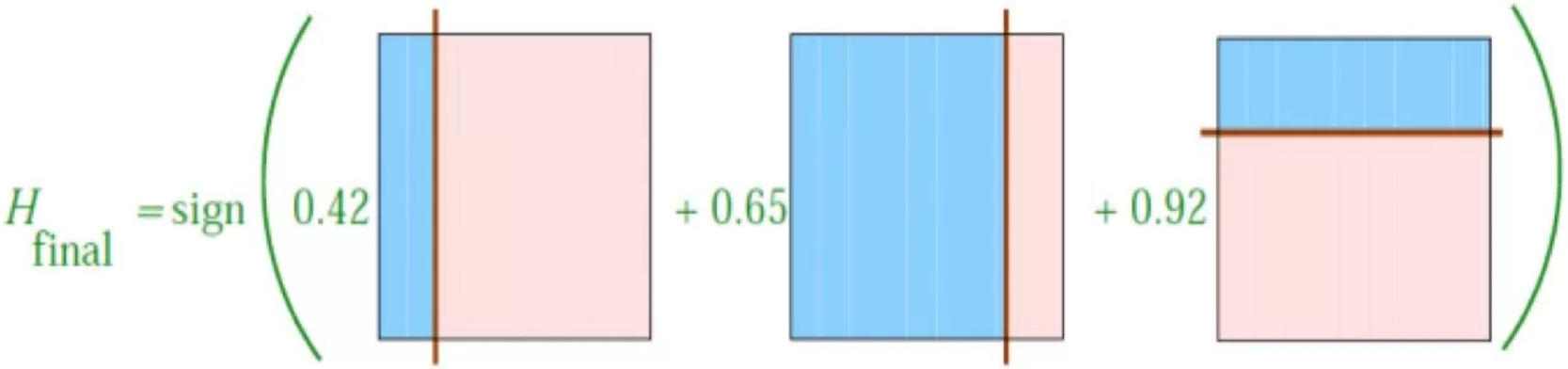
AdaBoost 图解 (4)

AdaBoost 图解： 第三个分类器 h_3 依据样本分布 $\mathcal{D}_3$ 进行训练，结果如图所示。

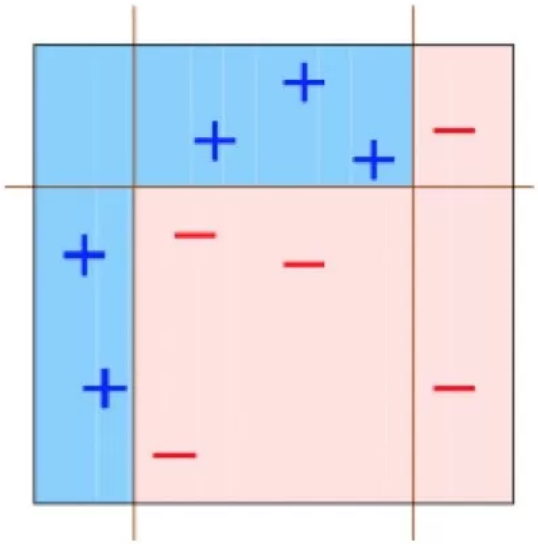


AdaBoost 图解 (5)

AdaBoost 图解： 将上述三个分类器通过线性加权，得到最终的分类器。



用这个最终的分类器对最初的样本进行分类，得到结果如下：



AdaBoost 例子 (1)

AdaBoost 例子： 给定下列样本，使用简单的基于阈值比较的基学习器，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 初始化数据权值分布 $w_{1i} = 1/10 = 0.1, , i = 1, 2, ..., 10.$

$$\mathcal{D}_1 = \{0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1\}.$$

（1）$t = 1$：基于 $(D, \mathcal{D}_1)$ 训练分类器 h_1，计算其错误率 ϵ_1 与权值 α_1，生成新样本分布 $\mathcal{D}_2$。

(a) 在权值分布为 $\mathcal{D}_1$ 的训练数据上，阈值 v 取 2.5 是分类错误率最低，故基分类器为

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

AdaBoost 例子 (2)

(b) $h_1(x)$ 在训练数据集上有 3 个误分类点，错误率为 $\epsilon_1 = P(h_1(x_i) \neq y_i) = 0.3$。

(c) 计算 $h_1(x)$ 的系数：$\alpha_1 = \frac{1}{2} log \frac{1-\epsilon_1}{\epsilon_1} = 0.4236$。

(d) 更新训练数据的权值分布，产生 $\mathcal{D}_2$。计算方法：$w_{2i} = \frac{w_{1i}}{Z_1} e^{-\alpha_1 y_i h_1(x_i)}$，
表中：$w'_{1i} = w_{1i} e^{-\alpha_1 y_i h_1(x_i)}$，$Z_1 = \sum_{i=1}^{10} w'_{1i} = 0.9166$

$$\mathcal{D}_2 = \{0.0715, 0.0715, 0.0715, 0.0715, 0.0715, 0.0715, 0.1666, 0.1666, 0.1666, 0.0715\}.$$

(f) 第一次集成的学习器：$H(x) = \text{sign}\,(f_1(x))$，其中 $f_1(x) = 0.4236 h_1(x)$。
分类器 $H(x)$ 在训练集上有 3 个误分类点。

x	0	1	2	3	4	5	6	7	8	9
y	+1	+1	+1	-1	-1	-1	+1	+1	+1	-1
w_{1i}	.1	.1	.1	.1	.1	.1	.1	.1	.1	.1
$h_1(x_i)$	+1	+1	+1	-1	-1	-1	-1	-1	-1	-1
w'_{1i}	.0655	.0655	.0655	.0655	.0655	.0655	.1527	.1527	.1527	.0655
w_{2i}	.0715	.0715	.0715	.0715	.0715	.0715	.1666	.1666	.1666	.0715
$f_1(x_i)$	.4236	.4236	.4236	-.4236	-.4236	-.4236	-.4236	-.4236	-.4236	-.4236
$H(x_i)$	+1	+1	+1	-1	-1	-1	-1	-1	-1	-1

AdaBoost 例子 (3)

（2）$t=2$：基于 $(D,\mathcal{D}_2)$ 训练分类器 h_2，计算其错误率 ϵ_2 与权值 α_2，生成新样本分布 $\mathcal{D}_3$。

(a) 在权值分布为 $\mathcal{D}_2$ 的训练数据上，阈值 v 取 8.5 的分类错误率最低，故基分类器为

$$h_2(x)=\begin{cases}+1, & x\leqslant 8.5\\ -1, & x>8.5\end{cases}$$

(b) $h_2(x)$ 在训练数据集上有 3 个误分类点，错误率为 $\epsilon_2=P(h_2(x_i)\neq y_i)=0.0715*3=0.2145$。

(c) 计算 $h_2(x)$ 的权值：$\alpha_2=\frac{1}{2}log\frac{1-\epsilon_2}{\epsilon_2}=0.6490$。

(d) 更新训练数据的权值分布，产生 $\mathcal{D}_3$。计算方法：$w_{3i}=\frac{w_{2i}}{Z_2}e^{-\alpha_2 y_i h_2(x_i)}$。

表中：$w'_{2i}=w_{2i}e^{-\alpha_2 y_i h_2(x_i)}$, $Z_2=\sum_{i=1}^{10}w'_{2i}=0.8213$

$$\mathcal{D}_3=\{0.0455,0.0455,0.0455,0.1667,0.1667,0.1667,0.1060,0.1060,0.1060,0.0455\}.$$

(f) 第二次集成的学习器：$H(x)=\text{sign}\,(f_2(x))$，其中 $f_2(x_i)=0.4236h_1(x)+0.6490h_x(2)$.
分类器 $H(x)$ 在训练数据集上有 3 个误分类点。

x	0	1	2	3	4	5	6	7	8	9
y	+1	+1	+1	-1	-1	-1	+1	+1	+1	-1
w_{2i}	.0715	.0715	.0715	.0715	.0715	.0715	.1666	.1666	.1666	.0715
$h_2(x_i)$	+1	+1	+1	+1	+1	+1	+1	+1	+1	-1
w'_{2i}	.0374	.0374	.0374	.1368	.1368	.1368	.0871	.0871	.0871	.0374
w_{3i}	.0455	.0455	.0455	.1667	.1667	.1667	.1060	.1060	.1060	.0455
$f_2(x_i)$	1.0726	1.0726	1.0726	0.2664	0.2664	0.2664	0.2664	0.2664	0.2664	-1.0726
$H(x_i)$	+1	+1	+1	+1	+1	+1	+1	+1	+1	-1

AdaBoost 例子 (4)

（3）$t = 3$：基于 $(D, \mathcal{D}_3)$ 训练分类器 h_3，计算其错误率 ϵ_3 与权值 α_3，生成新样本分布 $\mathcal{D}_4$。

(a) 在权值分布为 $\mathcal{D}_3$ 的训练数据上，阈值 v 取 5.5 的分类错误率最低，故基本分类器为

$$h_3(x) = \begin{cases} -1, & x \leqslant 5.5 \\ +1, & x > 5.5 \end{cases}$$

(b) $h_3(x)$ 在训练数据集上有 4 个误分类点，误差率为 $\epsilon_3 = P(h_3(x_i) \neq y_i) = 0.0455 * 4 = 0.1820.$

(c) 计算 $h_3(x)$ 的系数：$\alpha_3 = \frac{1}{2} log \frac{1-\epsilon_3}{\epsilon_3} = 0.7514.$

(d) 更新训练数据的权值分布，产生 $\mathcal{D}_4$。计算方法：$w_{4i} = \frac{w_{3i}}{Z_3} e^{-\alpha_3 y_i h_3(x_i)}$。

表中：$w'_{3i} = w_{3i} e^{-\alpha_3 y_i h_3(x_i)}$, $Z_3 = \sum_{i=1}^{10} w'_{3i} = 0.7718$

$$\mathcal{D}_4 = \{0.125, 0.125, 0.125, 0.102, 0.102, 0.102, 0.065, 0.065, 0.065, 0.125\}.$$

(f) 第三次集成的学习器 $H(x) = \text{sign}\,(f_3(x))$，其中 $f_3(x) = 0.4236h_1(x) + 0.6496h_2(x) + 0.7514h_3(x)$。分类器 $H(x)$ 在训练数据集上有 0 个误分类点。

x	0	1	2	3	4	5	6	7	8	9
y	+1	+1	+1	-1	-1	-1	+1	+1	+1	-1
w_{3i}	.0455	.0455	.0455	.1667	.1667	.1667	.1060	.1060	.1060	.0455
$h_3(x_i)$	-1	-1	-1	-1	-1	-1	+1	+1	+1	+1
w'_{3i}	.0965	.0965	.0965	.0786	.0786	.0786	.0500	.0500	.0500	.0965
w_{4i}	.125	.125	.125	.102	.102	.102	.065	.065	.065	.125
$f_3(x)$	.3212	.3212	.3212	-.485	-.485	-.485	1.0178	1.0178	1.0178	-.3213
$H(x)$	+1	+1	+1	-1	-1	-1	+1	+1	+1	-1

Boosting 算法与随机森林

欲得到泛化性能强的集成，集成中的个体学习器应尽可能相互独立；虽然“独立”在现实任务中无法做到，但可以设法使基学习器尽可能具有较大的差异。

实现方法：

- 对训练样本进行采样，产生出若干个不同的子集，再从每个数据子集中训练出一个基学习器。由于训练数据不同，我们获得的基学习器可望具有比较大的差异。
- 如果采样出的每个子集都完全不同，则每个基学习器只用到了一小部分训练数据，甚至不足以进行有效学习，无法确保产生出比较好的基学习器。
- 考虑使用相互有交叠的采样子集。

Bagging(1)

Bagging是并行式集成学习方法最著名的代表，其直接基于自助采样法。

自助采样法 (Bootstrap Sampling)

- **采用自助法生成采样集 D'**：每次随机从初始数据集 D 中挑出一个样本，将其拷贝放入采样集 D' 中；然后再将该样本**放回**初始数据集中 D；重复执行 m 次后，得到包含 m 个样本的采样集 D'。

  注意：D 中有一些样本可能在 D' 中多次出现，一些样本可能不在 D' 中出现。

- **采用自助法得到的测试集**：数据集 D 中未出现在 D' 中的所有样本作为测试集，即 $D - D'$。

显然，样本在 m 次采样中不出现在采样集 D' 的概率是：

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

即通过自助采样，初始数据集 D 中有约 36.8% 的样本未出现在采样数据集 D' 中。

Bagging(2)

Bagging 的基本流程：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。

结合预测输出的常用方法：

- 分类任务：投票法
- 回归任务：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 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_m, y_m)\}$;
基学习算法 $\mathfrak{L}$; 训练轮数 T;

过程：

1: **for** $t = 1, 2, ..., T$ **do**
2: $h_t = \mathfrak{L}(D, D_{bs})$; D_{bs} 是自助采样产生的样本分布。
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 能不经修改地用于多分类、回归等任务。
- Bagging 的计算复杂度与直接使用基学习算法训练一个学习器的复杂度同阶。
- 从偏差 -方差分解的角度看，Bagging 主要关注降低方差，在易受样本扰动的学习器上效果明显。

随机森林

随机森林 (Random Forest，简称 RF) 是 Bagging 的一个扩展变体。

传统决策树在选择划分属性时是在当前结点的属性集合 (假定有 d 个属性) 中选择一个最优属性。

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。

在 RF 中，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，然后再从这个子集中选择一个最优属性用于划分。

这里的参数 k 控制了随机性的引入程度：

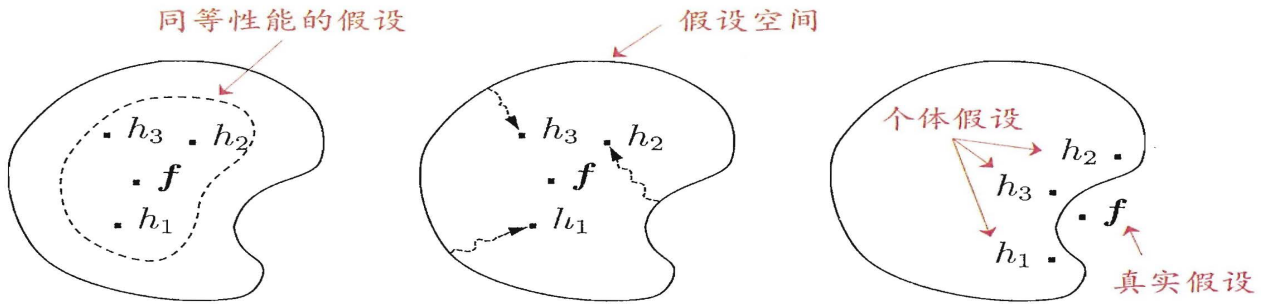
- 若令 $k = d$，则基决策树的构建与传统决策树相同；
- 若令 $k = 1$，则是随机选择一个属性用于划分；
- 一般情况下，推荐值 $k = \log_2 d$。

随机森林的训练效率常优于 Bagging，因为在个体决策树的构建过程中，Bagging 使用的是“确定型”决策树，在选择划分属性时要对结点的所有属性进行考察，而随机森林使用的“随机型”决策树则只需考察一个属性子集。

结合策略

结合策略的优点：

- **统计的方面**。由于学习任务的假设空间往往很大，可能有多个假设在训练集上达到同等性能，此时若使用单学习器可能因误选而导致泛化性能不佳，结合多个学习器则会减小这一风险；
- **计算的方面**。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，而通过多次运行之后进行结合，可降低陷入糟糕局部极小点的风险；
- **表示的方面**。某些学习任务的真实假设可能不在当前学习算法所考虑的假设空间中，此时若使用单学习器则肯定无效，而通过结合多个学习器，由于相应的假设空间有所扩大，有可能学得更好的近似。



(a) 统计的原因　(b) 计算的原因　(c) 表示的原因

学习器结合可能从三个方面带来好处 [Dietterich, 2000]

(来源：周志华，《机器学习》，清华大学出版社)

平均法

假定集成包含 T 个基学习器 $\{h_1, h_2, ..., h_T\}$，其中 h_i 在示例 $\boldsymbol{x}$ 上的输出为 $h_i(\boldsymbol{x})$。有几种对 h_i 进行结合的常见策略，如平均法、投票法、学习法等。

平均法（averaging）:

对数值型输出 $h_i(\boldsymbol{x}) \in \mathbb{R}$，最常见的结合策略是使用平均法。

- **简单平均法 (simple averaging)**

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

- **加权平均法 (weighted averaging)**

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

其中 ω_i 是个体学习器 h_i 的权重，通常要求 $\omega_i \geq 0$，$\sum_{i=1}^{T} \omega_i = 1$。

投票法 (1)

投票法 (voting)：

对分类任务来说，学习器 h_i 将从类别标记集合 $\{c_1, c_2, ..., c_N\}$ 中预测出一个标记，最常见的结合策略是使用投票法。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 $(h_i^1(\boldsymbol{x}), h_i^2(\boldsymbol{x}), ..., h_i^N(\boldsymbol{x}))$，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)**

$$H(\boldsymbol{x}) = \begin{cases} c_j, & \text{if } \sum_{i=1}^{T} h_i^j(\boldsymbol{x}) > 0.5 \sum_{k=1}^{N} \sum_{i=1}^{T} h_i^k(\boldsymbol{x}) \\ \text{reject}, & \text{otherwise} \end{cases} \tag{2}$$

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)**

$$H(\boldsymbol{x}) = c_{\arg\max_j \sum_{i=1}^{T} h_i^j(\boldsymbol{x})} \tag{3}$$

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。

- **加权投票法 (weighted voting)**

$$H(\boldsymbol{x}) = c_{\arg\max_j \sum_{i=1}^{T} \omega_i h_i^j(\boldsymbol{x})} \tag{4}$$

与加权平均法类似，ω_i 是 h_i 的权重，通常 $\omega_i \geq 0$，$\sum_{i=1}^{T} \omega_i = 1$。

投票法 (2)

投票法讨论：

标准的绝对多数投票法 (2)提供了“拒绝预测”选项，这在可靠性要求较高的学习任务中是一个很好的机制。若学习任务要求必须提供预测结果，则绝对多数投票法将退化为相对多数投票法。

式 (2)∼ (4)没有限制个体学习器输出值的类型。在现实任务中，不同类型个体学习器可能产生不同类型的 $h_i^j(\boldsymbol{x})$ 值，常见的有：

- **类标记：** $h_i^j \in \{0,1\}$，若 h_i 将样本 $\boldsymbol{x}$ 预测为类别 c_j 则取值为 1，否则为 0。使用类标记的投票亦称“硬投票” (hard voting)。
- **类概率：** $h_i^j \in [0,1]$，相当于对后验概率 $P(c_j|\boldsymbol{x})$ 的一个估计。使用类概率的投票亦称“软投票” (soft voting)。

注意：不同类型的 $h_i^j(\boldsymbol{x})$ 值不能混用。

学习法 (1)

学习法 (learning):

- 当训练数据很多时，一种更为强大的结合策略是使用“学习法”，即通过另一个学习器来进行结合。
- Stacking 是学习法的典型代表。这里我们把个体学习器称为初级学习器，用于结合的学习器称为次级学习器或元学习器 (meta-learner)。
- 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。在这个新数据集中，初级学习器的输出被当作样例输入特征，而初始样本的标记仍被当作样例标记.

学习法 (2)

Stacking 算法

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

过程:

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

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

- 使用初级学习算法 $\mathfrak{L}_t$ 产生初级学习器 h_t
- 生成次级训练集
- 在 D' 上用次级学习算法 $\mathfrak{L}$ 产生次级学习器 h'

学习法 (3)

Stacking 算法讨论：

在训练阶段，次级训练集是利用初级学习器产生的，若直接用初级学习器的训练集来产生次级训练集，则过拟合风险会比较大。利用交叉验证或留一法，训练初级学习器未使用的样本用于产生次级学习器的训练样本。

以 k 折交叉验证为例：

- 初始训练集 D 被随机划分为 k 个大小相似的集合 $D_1, D_2, ..., D_k$。令 D_j 和 $\bar{D}_j = D \backslash D_j$ 分别表示第 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 $\bar{D}_j$ 上使用第 t 个学习算法而得。
- 对 D_j 中每个样本 $\boldsymbol{x}_i$，另 $z_{it} = h_t^{(j)}(\boldsymbol{x}_i)$，则由 $\boldsymbol{x}_i$ 所产生的次级训练样本的示例部分为 $\boldsymbol{z}_i = (z_{i1}; z_{i2}; ...; z_{iT})$，标记部分为 y_i。
- 在整个交叉验证过程结束后，从这 T 个初级学习器产生的次级训练集是 $D' = \{(\boldsymbol{z}_i y_i)\}_{i=1}^m$，然后 D' 将用于训练次级学习器。

误差 -分歧分解

泛化能力强的集成由“好而不同”的个体学习器构建：即个体学习器的准确性越高、多样性越大，则集成的泛化能力越强。

简单分析： 对回归学习任务 $f:\mathbb{R}^d \mapsto \mathbb{R}$，个体学习器 $h_1, h_2, ..., h_T$ 通过加权平均法结合产生集成学习器 $H(\boldsymbol{x}) = \sum_{i=1}^{T} \omega_i h_i(\boldsymbol{x})$。

分歧： 定义 h_i 和 H 的分歧分别为：

$$\begin{aligned} A(h_x|\boldsymbol{x}) &= (h_i(\boldsymbol{x}) - H(\boldsymbol{x}))^2 \\ \overline{A}(h|\boldsymbol{x}) &= \sum_{i=1}^{T} \omega_i A(h_i|\boldsymbol{x}) \end{aligned}$$

这里的分歧项表征了个体学习器与集成学习器之间的不一致性，在一定程度上反映了个体学习器的多样性。

误差： h_i 和 H 的平方误差分别为：

$$\begin{aligned} E(h_i|\boldsymbol{x}) &= (f(\boldsymbol{x}) - h_i(\boldsymbol{x}))^2 \\ E(H|\boldsymbol{x}) &= (f(\boldsymbol{x}) - H(\boldsymbol{x}))^2 \end{aligned}$$

定义个体学习器误差的加权平均值为：

$$\overline{E}(h|\boldsymbol{x}) = \sum_{i=1}^{T} \omega_i E(h_i|\boldsymbol{x}).$$

误差 -分歧分解 (error-ambiguity decomposition)：

$$E(h|\boldsymbol{x}) = \overline{E}(h|\boldsymbol{x}) - \overline{A}(h|\boldsymbol{x})$$

上式表明，个体学习器的准确性越高、多样性越大，则集成越好。

多样性度量 (1)

多样性度量 (diversity measure)：用于度量集成中个体分类器的多样性，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。

给定数据集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_m, y_m)\}$，对二分类任务，$y_i \in \{-1, +1\}$，分类器 h_i 与 h_j 的预测结果列联表 (contingency table) 为

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

其中，a 表示 h_i 与 h_j 均预测为正类的样本数目；b、c、d 含义由此类推；$a + b + c + d = m$。

多样性度量 (2)

常见的多样性度量：

- **不合度量：** dis_{ij} 的值域为 $[0, 1]$，值越大则多样性越大。

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

- **相关系数：** ρ_{ij} 的值域为 $[-1, 1]$，若 h_i 与 h_j 无关，则值为 0；若 h_i 与 h_j 正相关，则值为正，否则为负。

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

- **Q-统计量：** Q_{ij} 与相关系数 ρ_{ij} 的符号相同，且 $|Q_{ij}| \leq |\rho_{ij}|$。

$$Q_{ij} = \frac{ad - bc}{ad + bc}$$

- **κ-统计量：** 若分类器 h_i 与 h_j 在数据集 D 上完全一致，则 $\kappa = 1$；若它们仅是偶然达成一致，则 $\kappa = 0$。κ 通常为非负值，仅在 h_i 与 h_j 达成一致的概率甚至低于偶然性的情况下取负值。

$$\kappa = \frac{p_1 - p_2}{1 - p_2}, \quad p_1 = \frac{a + d}{m}, \quad p_2 = \frac{(a + b)(a + c) + (c + d)(b + d)}{m^2}$$

其中，p_1 是两个分类器取得一致的概率，p_2 是两个分类器偶然达成一致的概率。

多样性增强

多样性增强的一般思路：在学习过程中引入随机性，常见做法主要是对数据样本、输入属性、输出表示、算法参数进行扰动。

- **数据样本扰动：**给定初始数据集，可从中产生出不同的数据子集，再利用不同的数据子集训练出不同的个体学习器。数据样本扰动通常是基于采样法。例如 Bagging 中使用的自助采样，在 AdaBoost 中使用的序列采样。
- **输入属性扰动：**训练样本通常由一组属性描述，不同的“子空间”（即属性子集）提供了观察数据的不同视角。从不同子空间训练出的个体学习器必然有所不同。著名的随机子空间算法就依赖于输入属性扰动，该算法从初始属性集中抽取出若干个属性子集，再基于每个属性子集训练一个基学习器。
- **输出表示扰动：**基本思路是对输出表示进行操纵以增强多样性。可对训练样本的类标记稍作变动，如“翻转法”随机改变一些训练样本的标记；也可对输出表示进行转化，如“输出调制法”将分类输出转化为回归输出后构建个体学习器；还可将原任务拆解为多个可同时求解的子任务。
- **算法参数扰动：**基学习算法一般都有参数需进行设置，例如神经网络的隐层神经元数、初始连接权值等，通过随机设置不同的参数，产生差别较大的个体学习器。

小结

本章主要讲解了如何利用简单学习器构造集成学习器，主要内容包括：

- 个体与集成的基本概念。
- Boosting 算法：个体学习器间存在强依赖关系、通过串行生成的序列化方法进行集成学习。
- Bagging 算法：个体学习器间不存在强依赖关系、通过同时生成的并行化方法进行集成学习。
- 结合策略：如何结合个体学习器，包括平均法，投票法，学习法等。
- 多样性：个体学习器的准确性越高、多样性越大，则集成的泛化能力越强。