集成学习研究的核心：如何产生并结合“好而不同”的个体学习器。根据个体学习器的生成方式，集成学习大致分为两类：一是，个体学习器间存在较强依赖关系、必须串行生成的序列化方法，boosting；二是，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，bagging和随即森林。

1.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 T, 最终将这 T 个基学习器进行加权结合.

代表：Adaboost(Adaptive boosting)

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

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

图 8.3 AdaBoost算法

公式推导可见《机器学习》P174

优点：泛化错误率低，易编码，可以应用在大部分分类器上，无参数调整

缺点：对离群点敏感

适用数据类型：数值型和标称型数据

提升树：以决策树为基函数的提升方法，boosting tree

单层决策树（decision stump,也称决策树桩）

2.Bagging

并行式集成学习　自助采样法

结合时，对分类任务简单投票，对回归任务简单平均

扩展：随机森林——属性选择的随机性

随机森林简单容易实现，计算开销小，在很多现实任务中展现出强大的性能，对bagging只做了小改动，但随机森林中基学习器的多样性不仅来自样本扰动，还来自属性扰动，使得最终集成的泛化性能可通过个体学习器之间差异度的增加而进一步提升。

随机森林顾名思义，是用随机的方式建立一个森林，森林里面有很多的决策树组成，随机森林的每一棵决策树之间是没有关联的。在得到森林之后，当有一个新的输入样本进入的时候，就让森林中的每一棵决策树分别进行一下判断，看看这个样本应该属于哪一类（对于分类算法），然后看看哪一类被选择最多，就预测这个样本为那一类。

在建立每一棵决策树的过程中，有两点需要注意 - 采样与完全分裂。首先是**两个随机采样**的过程，random forest对输入的数据要进行行、列的采样。对于行采样，采用有放回的方式，也就是在采样得到的样本集合中，可能有重复的样本。假设输入样本为N个，那么采样的样本也为N个。这样使得在训练的时候，**每一棵树的输入样本都不是全部的样本，使得相对不容易出现**over-fitting。然后进行列采样，**从M个feature中，选择m个**(m << M)。之后就是对采样之后的数据使用完全分裂的方式建立出决策树，这样决策树的某一个叶子节点要么是无法继续分裂的，要么里面的所有样本的都是指向的同一个分类。一般很多的决策树算法都一个重要的步骤 - 剪枝，但是这里不这样干，由于之前的两个随机采样的过程保证了随机性，所以就算**不剪枝，也不会出现**over-fitting。

按这种算法得到的随机森林中的每一棵都是很弱的，但是大家组合起来就很厉害了。我觉得可以这样比喻随机森林算法：每一棵决策树就

是一个精通于某一个窄领域的专家（因为我们从M个feature中选择m让每一棵决策树进行学习），这样在随机森林中就有了很多个精通不同领域的专家，对一个新的问题（新的输入数据），可以用不同的角度去看待它，最终由各个专家，投票得到结果。

3.bagging是减少variance，而boosting是减少bias

Bagging对样本重采样，对每一重采样得到的子样本集训练一个模型，最后取平均。由于子样本集的相似性以及使用的是同种模型，因此各模型有近似相等的bias和variance（事实上，各模型的分布也近似相同，但不独立）。由于

$$E[\frac{\sum X_i}{n}] = E[X_i]$$

E[\frac{\sum X_i}{n}]=E[X_i]

，所以bagging后的bias和单个子模型的接近，一般来说不能显著降低bias。另一方面，若各子模型独立，则有

$$Var(\frac{\sum X_i}{n}) = \frac{Var(X_i)}{n}$$

Var(\frac{\sum X_i}{n})=\frac{Var(X_i)}{n}

，此时可以显著降低variance。若各子模型完全相同，则

$$Var(\frac{\sum X_i}{n}) = Var(X_i)$$

Var(\frac{\sum X_i}{n})=Var(X_i)

，此时不会降低variance。bagging方法得到的各子模型是有一定相关性的，属于上面两个极端状况的中间态，因此可以一定程度降低variance。为了进一步降低variance，Random forest通过随机选取变量子集做拟合的方式de-correlated了各子模型（树），使得variance进一步降低。

boosting从优化角度来看，是用forward-stagewise这种贪心法去最小化损失函数

$$L(y, \sum_i a_i f_i(x))$$

L(y, \sum_i a_i f_i(x))

。例如，常见的AdaBoost即等价于用这种方法最小化exponential loss：

$$L(y, f(x)) = exp(-yf(x))$$

L(y,f(x))=exp(-yf(x))

。所谓forward-stagewise，就是在迭代的第n步，求解新的子模型f(x)及步长a（或者叫组合系数），来最小化

$$L(y, f_{n-1}(x) + af(x))$$

L(y,f_{n-1}(x)+af(x))

，这里

$$f_{n-1}(x)$$

f_{n-1}(x)

是前n-1步得到的子模型的和。因此boosting是在sequential地最小化损失函数，其bias自然逐步下降。但由于是采取这种sequential、adaptive的策略，各子模型之间是强相关的，于是子模型之和并不能显著降低variance。所以说boosting主要还是靠降低bias来提升预测精度。