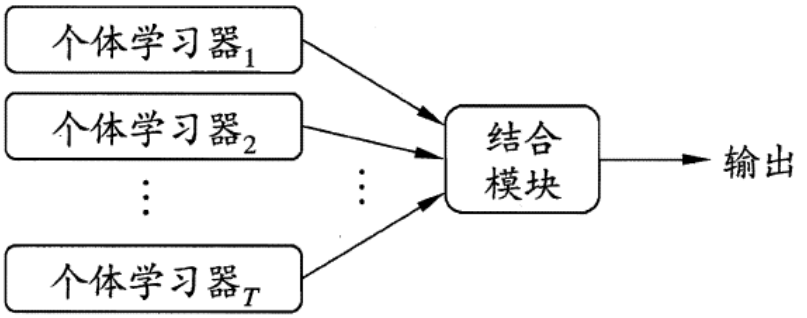
8、集成学习

顾名思义，集成学习（ensemble learning）指的是将多个学习器进行有效地结合，组建一个“学习器委员会”，其中每个学习器担任委员会成员并行使投票表决权，使得委员会最后的决定更能够四方造福普度众生~...~，即其泛化性能要能优于其中任何一个学习器。

8.1 个体与集成

集成学习的基本结构为：先产生一组个体学习器，再使用某种策略将它们结合在一起。集成模型如下图所示：



在上图的集成模型中，若个体学习器都属于同一类别，例如都是决策树或都是神经网络，则称该集成为同质的（homogeneous）;若个体学习器包含多种类型的学习算法，例如既有决策树又有神经网络，则称该集成为异质的（heterogenous）。

同质集成：个体学习器称为“基学习器”（base learner），对应的学习算法为“基学习算法”（base learning algorithm）。**异质集成**：个体学习器称为“组件学习器”（component learner）或直称为“个体学习器”。

上面我们已经提到要让集成起来的泛化性能比单个学习器都要好，虽说团结力量大但也有木桶短板理论调皮捣蛋，那如何做到呢？这就引出了集成学习的两个重要概念：**准确性**和**多样性**（diversity）。准确性指的是个体学习器不能太差，要有一定的准确度；多样性则是个体学习器之间的输出要具有差异性。通过下面的这三个例子可以很容易看出这一点，准确度较高，差异度也较高，可以较好地提升集成性能。

准确度较高，差异度较高

	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

准确度较高，差异度较低

	测试例1	测试例2	测试例3
h_1	√	√	×
h_2	√	√	×
h_3	√	√	×
集成	√	√	×

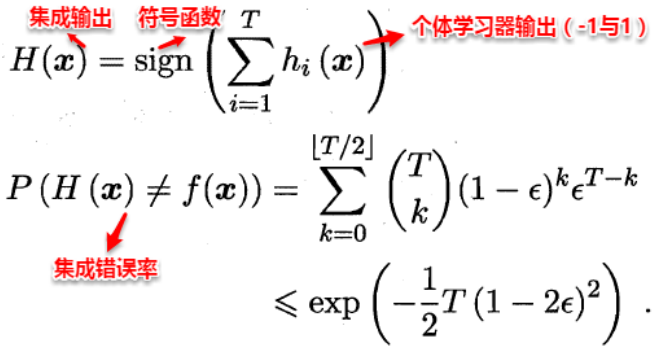
(b) 集成不起作用

准确度较低，差异度较高

	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

现在考虑二分类的简单情形，假设基分类器之间相互独立（能提供较高的差异度），且错误率相等为 ε，则可以将集成器的预测看做一个伯努利实验，易知当所有基分类器中不足一半预测正确的情况下，集成器预测错误，所以集成器的错误率可以计算为：



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

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

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

此时，集成器错误率随着基分类器的个数的增加呈指数下降，但前提是基分类器之间相互独立，在实际情形中显然是不可能的，假设训练有 A 和 B 两个分类器，对于某个测试样本，显然满足：P（A=1 | B=1）> P（A=1），因为 A 和 B 为了解决相同的问题而训练，因此在预测新样本时存在着很大的联系。因此，**个体学习器的“准确性”和“差异性”本身就是一对矛盾的变量**，准确性高意味着牺牲多样性，所以产生“好而不同”的个体学习器正是集成学习研究的核心。现阶段有三种主流的集成学习方法：Boosting、Bagging 以及随机森林（Random Forest），接下来将进行逐一介绍。

8.2 Boosting

Boosting 是一种串行的工作机制，即个体学习器的训练存在依赖关系，必须一步一步序列化进行。其基本思想是：增加前一个基学习器在训练训练过程中预测错误样本的权重，使得后续基学习器更加关注这些打标错误的训练样本，尽可能纠正这些错误，一直向下串行直至产生需要的 T 个基学习器，Boosting 最终对这 T 个学习器进行加权结合，产生学习器委员会。

Boosting 族算法最著名、使用最为广泛的就是 AdaBoost，因此下面主要是对 AdaBoost 算法进行介绍。AdaBoost 使用的是**指数损失函数**，因此 AdaBoost 的权值与样本分布的更新都是**围绕着最小化指数损失函数**进行的。看到这里回想一下之前的机器学习算法，不难发现机器学习的大部分带参模型只是改变了最优化目标中的损失函数：如果是 Square loss，那就是最小二乘了；如果是 Hinge Loss，那就是著名的 SVM 了；如果是 log-Loss，那就是 Logistic Regression 了。

定义基学习器的集成为加权结合，则有：

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

AdaBoost 算法的指数损失函数定义为：

$$loss_{\exp}(h) = \mathbb{E}_{x\sim\mathcal{D},y}[e^{-yh(x)}]$$ → 打标正确为负 打标错误为正

具体说来，整个 Adaboost 迭代算法分为 3 步：

- 初始化训练数据的权值分布。如果有 N 个样本，则每一个训练样本最开始时都被赋予相同的权值：1/N。
- 训练弱分类器。具体训练过程中，如果某个样本点已经被准确地分类，那么在构造下一个训练集中，它的权值就被降低；相反，如果某个样本点没有被准确地分类，那么它的权值就得到提高。然后，权值更新过的样本集被用于训练下一个分类器，整个训练过程如此迭代地进行下去。
- 将各个训练得到的弱分类器组合成强分类器。各个弱分类器的训练过程结束后，加大分类误差率小的弱分类器的权重，使其在最终的分类函数中起着较大的决定作用，而降低分类误差率大的弱分类器的权重，使其在最终的分类函数中起着较小的决定作用。

整个 AdaBoost 的算法流程如下所示：

Input: Data set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
Base learning algorithm L;
Number of learning rounds T.

Process:

1. $\mathcal{D}_1(i) = 1/m$. % Initialize the weight distribution
2. **for** $t = 1, \cdots, T$:
3. $h_t = L(D, \mathcal{D}_t)$; % Train a learner h_t from D using distribution $\mathcal{D}_t$
4. $\epsilon_t = \Pr_{x\sim\mathcal{D}_t,y}\boldsymbol{I}[h_t(\boldsymbol{x}) \neq y]$; % Measure the error of h_t
5. **if** $\epsilon_t > 0.5$ **then break**
6. $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$; % Determine the weight of h_t （1）计算基学习器权重
7. $\mathcal{D}_{t+1}(i) = \frac{\mathcal{D}_t(i)}{Z_t} \times \begin{cases} \exp(-\alpha_t) & \text{if } h_t(\boldsymbol{x}_i) = y_i \\ \exp(\alpha_t) & \text{if } h_t(\boldsymbol{x}_i) \neq y_i \end{cases}$ （2）样本权重分布更新
$= \frac{\mathcal{D}_t(i)\exp(-\alpha_t y_i h_t(\boldsymbol{x}_i))}{Z_t}$ % Update the distribution, where
% Z_t is a normalization factor which
% enables $\mathcal{D}_{t+1}$ to be distribution
8. **end**

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

可以看出：AdaBoost 的核心步骤就是**计算基学习器权重和样本权重分布**，那为何是上述的计算公式呢？这就涉及到了我们之前为什么说大部分带参机器学习算法只是改变了损失函数，就是因为大部分模型的**参数都是通过最优化损失函数（可能还加个规则项）而计算（梯度下降，坐标下降等）得到，这里正是通过最优化指数损失函数从而得到这两个参数的计算公式**，具体的推导过程此处不进行展开。

Boosting 算法要求基学习器能对特定分布的数据进行学习，即每次都更新样本分布权重，这里书上提到了两种方法："重赋权法"（re-weighting）和"重采样法"（re-sampling），书上的解释有些晦涩，这里进行展开一下：

重赋权法：对每个样本附加一个权重，这时涉及到样本属性与标签的计算，都需要乘上一个权值。
重采样法：对于一些无法接受带权样本的及学习算法，适合用"重采样法"进行处理。方法大致过程是，根据各个样本的权重，对训练数据进行重采样，初始时样本权重一样，每个样本被采样到的概率一致，每次从 N 个原始的训练样本中按照权重有放回采样 N 个样本作为训练集，然后计算训练集错误率，然后调整权重，重复采样，集成多个基学习器。
从偏差-方差分解来看：Boosting 算法主要关注于降低偏差，每轮的迭代都关注于训练过程中预测错误的样本，将弱学习提升为强学习器。从 AdaBoost 的算法流程来看，标准的 AdaBoost 只适用于二分类问题。在此，当选为数据挖掘十大算法之一的 AdaBoost 介绍到这里，能够当选正是说明这个算法十分婀娜多姿，背后的数学证明和推导充分证明了这一点，限于篇幅不再继续展开。

8.3 Bagging 与 Random Forest

相比之下，Bagging 与随机森林算法就简洁了许多，上面已经提到产生"好而不同"的个体学习器是集成学习研究的核心，即在保证基学习器准确性的同时增加基学习器之间的多样性。而这两种算法的基本思（tao）想（lu）都是通过"自助采样"的方法来增加多样性。

8.3.1 Bagging

Bagging 是一种并行式的集成学习方法，即基学习器的训练之间没有前后顺序可以同时进行，Bagging 使用"有放回"采样的方式选取训练集，对于包含 m 个样本的训练集，进行 m 次有放回的随机采样操作，从而得到 m 个样本的采样集，这样训练集中有接近 36.8%的样本没有被采到。按照相同的方式重复进行，我们就可以采集到 T 个包含 m 个样本的数据集，从而训练出 T 个基学习器，最终对这 T 个基学习器的输出进行结合。

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

Bagging 算法的流程如下所示：

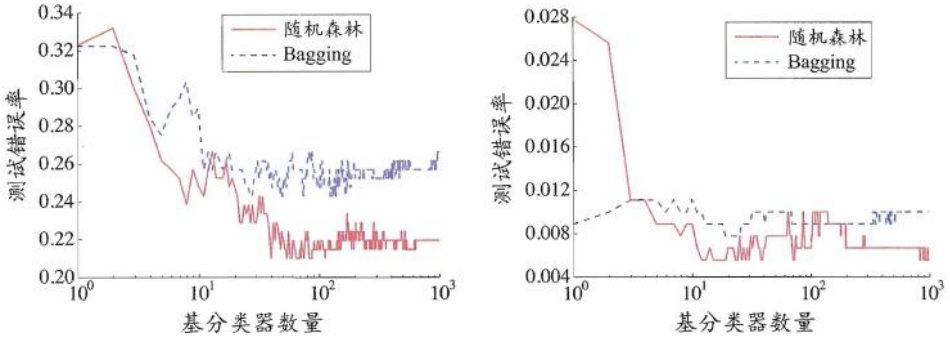
输入： 训练集 $D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\ldots,(\boldsymbol{x}_m,y_m)\}$;
基学习算法 $\mathfrak{L}$;
训练轮数 T.
过程：
1: **for** $t=1,2,\ldots,T$ **do**
2: $h_t=\mathfrak{L}(D,\mathcal{D}_{bs})$
3: **end for**
输出： $H(\boldsymbol{x})=\arg\max_{y\in\mathcal{Y}}\sum_{t=1}^{T}\mathbb{I}(h_t(\boldsymbol{x})=y)$

可以看出 Bagging 主要通过**样本的扰动**来增加基学习器之间的多样性，因此 Bagging 的基学习器应为那些对训练集十分敏感的不稳定学习算法，例如：神经网络与决策树等。从偏差-方差分解来看，Bagging 算法主要关注**于降低方差，即通过多次重复训练提高稳定性**。不同于 AdaBoost 的是，Bagging 可以十分简单地移植到多分类、回归等问题。总的说起来则是：**AdaBoost 关注于降低偏差，而 Bagging 关注于降低方差。**

8.3.2 随机森林

随机森林（Random Forest）是 Bagging 的一个拓展体，它的基学习器固定为决策树，多棵树也就组成了森林，而"随机"则在于选择划分属性的随机，随机森林在训练基学习器时，也采用有放回采样的方式添加样本扰动，同时它还引入了一种**属性扰动**，即在基决策树的训练过程中，在选择划分属性时，RF 先从候选属性集中随机挑选出一个包含 K 个属性的子集，再从这个子集中选择最优划分属性，一般推荐 K=log2（d）。

这样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，从而进一步提升了基学习器之间的差异度。相比决策树的 Bagging 集成，随机森林的起始性能较差（由于属性扰动，基决策树的准确度有所下降），但随着基学习器数目的增多，随机森林往往会收敛到更低的泛化误差。同时不同于 Bagging 中决策树从所有属性集中选择最优划分属性，随机森林只在属性集的一个子集中选择划分属性，因此训练效率更高。



(a) glass 数据集　　(b) auto-mpg 数据集

8.4 结合策略

结合策略指的是在训练好基学习器后，如何将这些基学习器的输出结合起来产生集成模型的最终输出，下面将介绍一些常用的结合策略：

8.4.1 平均法（回归问题）

- 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)

$$H(\boldsymbol{x}) = \sum_{i=1}^{T} w_i h_i(\boldsymbol{x}) \,. \qquad \text{通常要求 } w_i \geqslant 0, \sum_{i=1}^{T} w_i = 1.$$

易知简单平均法是加权平均法的一种特例，加权平均法可以认为是集成学习研究的基本出发点。由于各个基学习器的权值在训练中得出，**一般而言，在个体学习器性能相差较大时宜使用加权平均法，在个体学习器性能相差较小时宜使用简单平均法。**

8.4.2 投票法（分类问题）

- 绝对多数投票法(majority voting)　必须要占一半以上

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

- 相对多数投票法(plurality voting)　最多票数即可

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

- 加权投票法(weighted voting)

$$H(\boldsymbol{x}) = c_{\arg\max_j \sum_{i=1}^{T} w_i h_i^j(\boldsymbol{x})} \,. \quad \text{通常 } w_i \geqslant 0, \sum_{i=1}^{T} w_i = 1.$$

绝对多数投票法（majority voting）提供了拒绝选项，这在可靠性要求很高的学习任务中是一个很好的机制。同时，对于分类任务，各个基学习器的输出值有两种类型，分别为类标记和类概率。

- 类标记: $h_i^j(\boldsymbol{x}) \in \{0,1\}$, 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: $h_i^j(\boldsymbol{x}) \in [0,1]$, 相当于对后验概率 $P(c_j \mid \boldsymbol{x})$ 的一个估计. 使用类概率的投票亦称“软投票”(soft voting).

一些在产生类别标记的同时也生成置信度的学习器，置信度可转化为类概率使用，**一般基于类概率进行结合往往比基于类标记进行结合的效果更好，**需要注意的是对于异质集成，其类概率不能直接进行比较，此时需要将类概率转化为类标记输出，然后再投票。

8.4.3 学习法

学习法是一种更高级的结合策略，即学习出一种“投票”的学习器，Stacking 是学习法的典型代表。Stacking 的基本思想是：首先训练出 T 个基学习器，对于一个样本它们会产生 T 个输出，将这 T 个基学习器的输出与该样本的真实标记作为新的样本，m 个样本就会产生一个 m*T 的样本集，来训练一个新的“投票”学习器。投票学习器的输入属性与学习算法对 Stacking 集成的泛化性能有很大的影响，书中已经提到：**投票学习器采用类概率作为输入属性，选用多响应线性回归（MLR）一般会产生较好的效果。**

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: $h_t=\mathfrak{L}_t(D)$;
3: **end for**
4: $D'=\varnothing$;
5: **for** $i=1,2,\ldots,m$ **do**
6: **for** $t=1,2,\ldots,T$ **do**
7: $z_{it}=h_t(\boldsymbol{x}_i)$;
8: **end for**
9: $D'=D'\cup((z_{i1},z_{i2},\ldots,z_{iT}),y_i)$;
10: **end for**
11: $h'=\mathfrak{L}(D')$;
输出： $H(\boldsymbol{x})=h'(h_1(\boldsymbol{x}),h_2(\boldsymbol{x}),\ldots,h_T(\boldsymbol{x}))$

8.5 多样性（diversity）

在集成学习中，基学习器之间的多样性是影响集成器泛化性能的重要因素。因此增加多样性对于集成学习研究十分重要，一般的思路是在学习过程中引入随机性，常见的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，即利用具有差异的数据集来训练不同的基学习器。例如：有放回自助采样法，但此类做法只对那些不稳定学习算法十分有效，例如：决策树和神经网络等，训练集的稍微改变能导致学习器的显著变动。 **输入属性扰动**，即随机选取原空间的一个子空间来训练基学习器。例如：随机森林，从初始属性集中抽取子集，再基于每个子集来训练基学习器。但若训练集只包含少量属性，则不宜使用属性扰动。 **输出表示扰动**，此类做法可对训练样本的类标稍作变动，或对基学习器的输出进行转化。 **算法参数扰动**，通过随机设置不同的参数，例如：神经网络中，随机初始化权重与随机设置隐含层节点数。
在此，集成学习就介绍完毕，看到这里，大家也会发现集成学习实质上是一种通用框架，可以使用任何一种基学习器，从而改进单个学习器的泛化性能。