集成学习方法

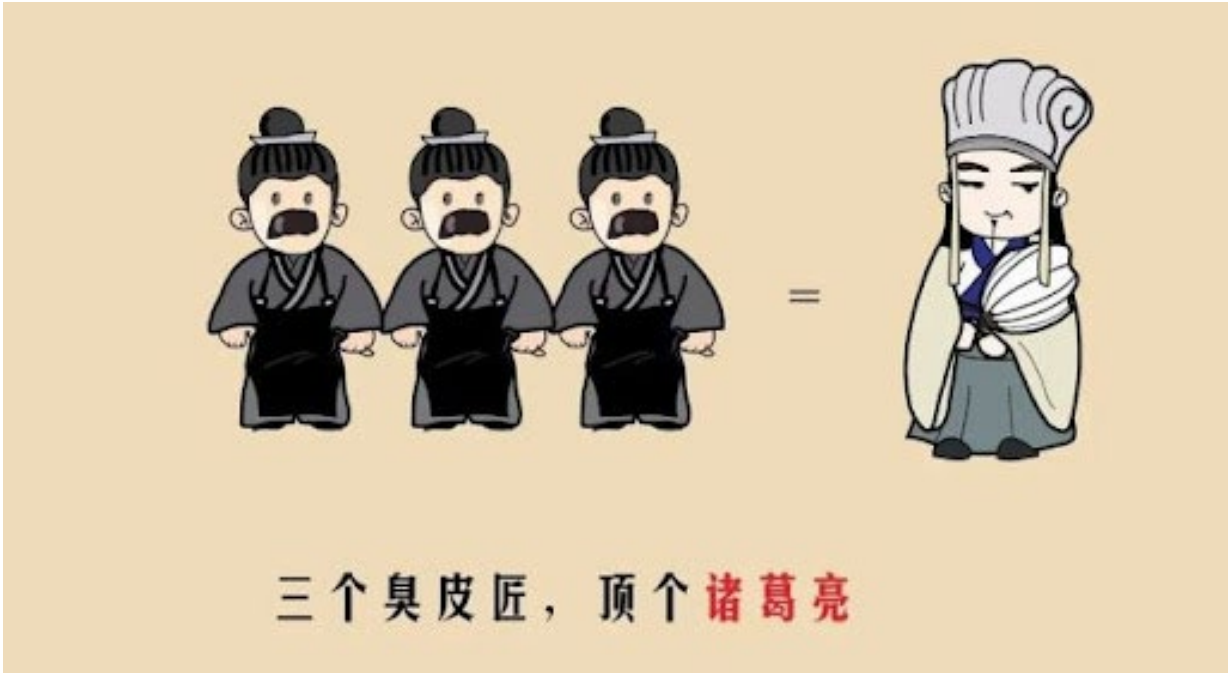
Classification and Regression

集成学习

- 集成（Ensemble）学习思想
  - 在监督学习算法中，我们的目标是学习出一个稳定的且在各个方面表现都较好的模型，但实际情况往往不这么理想，有时我们只能得到多个有偏好的模型（在某些方面表现的比较好，但泛化能力比较差，例如前面讲的决策树模型），称之为**弱学习器**。
  - 解决这一问题的一个朴素的思想是：使用一系列弱学习器进行学习，并按照某种规则把各个学习的结果进行整合，从而构成一个强大的整体，获得比单个学习器显著优越的泛化性能，该整体可以解决更复杂的问题，这就是**集成学习**。
  - 也被称为“多分类器系统”或“基于委员会的学习”

集成学习

- 集成（Ensemble）思想的本质



即便某一个弱分类器得到了错误的预测，其他的弱分类器也可以将错误纠正回来，例如：学习算法A在a情况下失效，学习算法B在b情况下失效，那么在a情况下可以用B算法，在b情况下可以用A算法解决。这说明通过某种合适的方式把各种算法整合起来，可以提高准确率。

集成学习方法

- 不是一种单独的机器学习算法，而更像是一种优化策略
- 因为单个机器学习模型所能解决的问题有限，泛化能力差，但是通过构建组合多个基本的学习器来完成学习任务往往能够获得奇效
- 实现集成学习需要解决两个问题：
  - 怎样获得不同的弱学习器？
  - 怎样整合这些弱学习器？

集成学习方法

- 怎样获得不同的弱学习器？
  - 使用不同的弱学习算法得到不同的基本学习器

    eg: Linear、Ridge、Lasso、Logistic、ID3、C4.5、CART、SVM、KNN
  - 使用相同的弱学习算法，但用不同的参数
  - 相同输入对象的不同表示凸显事物不同的特征
  - 使用不同的训练集

    随机抽样（Bagging）

    改变训练集样本的权重分布（Boosting）

集成学习方法

- 怎样整合这些弱学习器？
  - **序列化的方法（多级组合）**

    一种串行结构，其中下一个学习器根据前一个学习器的学习情况进行训练，个体学习器的产生相互依赖，比如当前学习器的产生依赖上一个学习器的参数，最终将多个学习器的输出结果进行整合，代表性方法是Boosting
  - **并行化方法（多专家组合）**

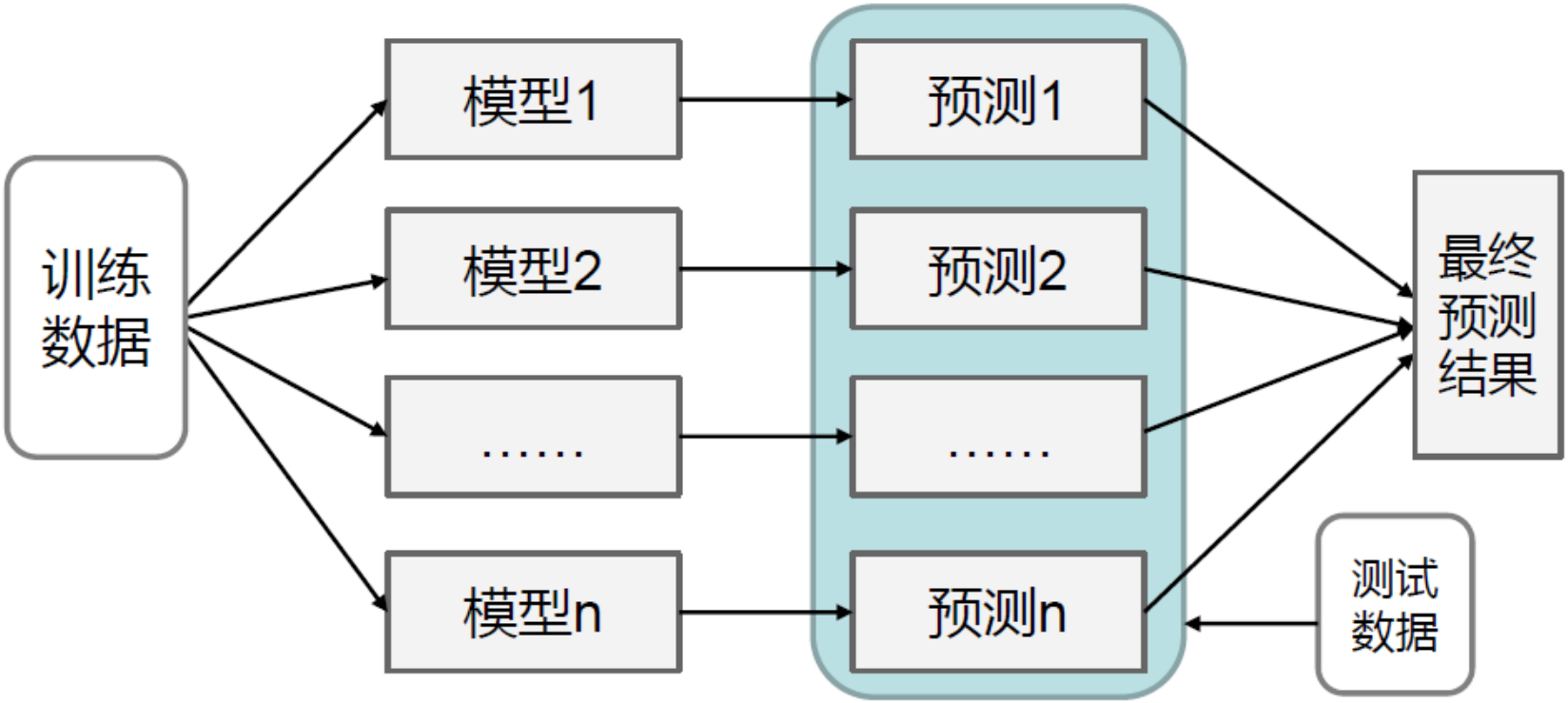
    一种并行结构，个体学习器间不存在强依赖关系，可以并行化同时产生，最后将多个学习器的输出结果整合，代表性方法是Bagging

集成学习方法

- Bagging
  - 也称为自举汇聚法 (boostrap aggregating)
  - 采用自助法(boostrap)即有放回的抽样
  - 从原始数据集抽样S次后得到S个新数据集
  - 每个数据集的大小相等
  - 训练数据中允许存在重复数据
  - 每个数据集都是通过在原始数据集中随机选择样本来进行替换而得到的。
  - S个数据集建好之后，将某个学习算法分别作用于每个数据集就得到S个基本学习器。
  - 代表性算法:随机森林(Random Forest)

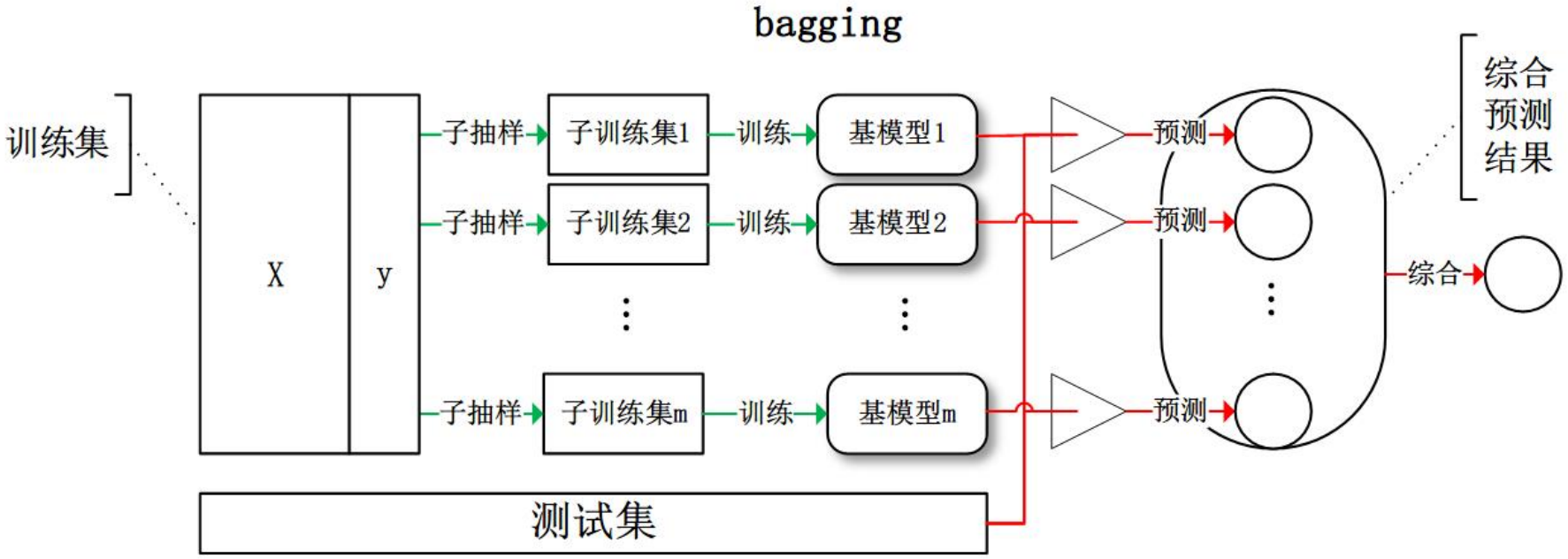
集成学习方法

- Bagging
  - 也称为自举汇聚法 (boostrap aggregating)



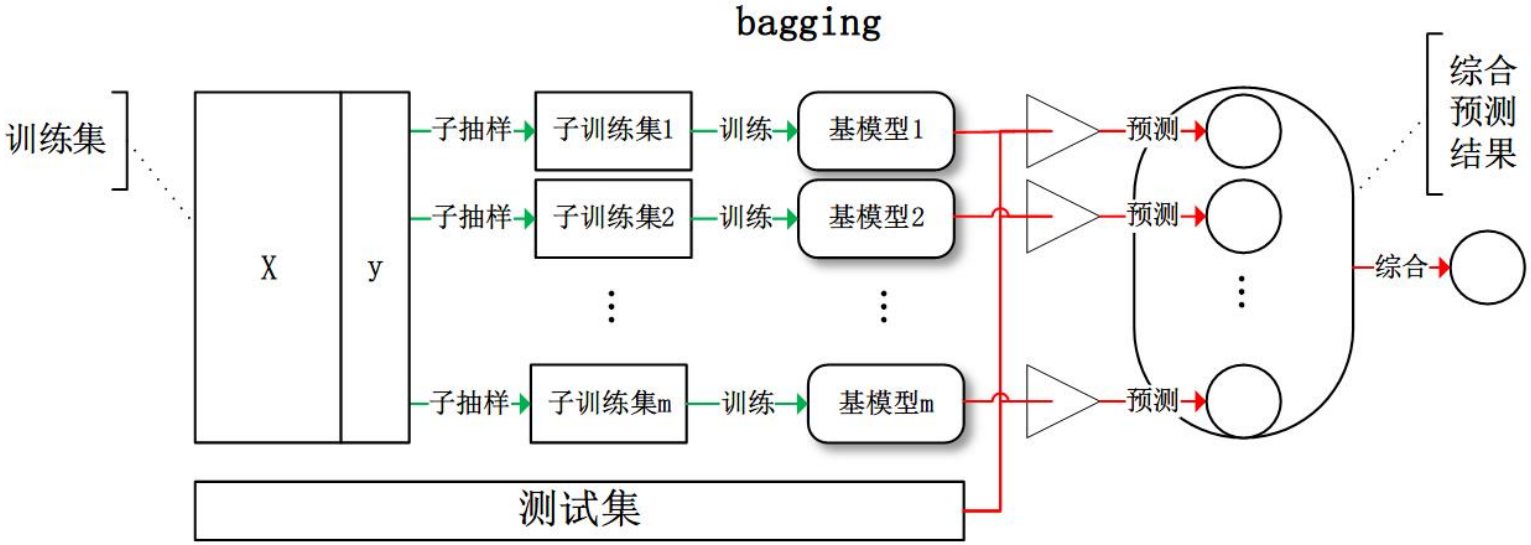
集成学习方法

- Bagging
  - 从训练集中进行子抽样组成每个基模型所需要的子训练集，对所有基模型预测的结果进行综合，从而产生最终的预测结果



集成学习方法

- Bagging的结合策略
  - 对于分类问题，通常使用简单投票法，得到最多票数的类别或者类别之一为最终的模型输出
  - 对于回归问题，通常使用简单平均法，对 T个弱学习器得到的回归结果进行算术平均得到最终的模型输出



集成学习方法

- Bagging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：

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

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

图 8.5 Bagging 算法

集成学习方法

- Bagging总结
  - Bagging通过多个基学习器输出结果的平均，降低了集成后整体的方差，改善了泛化误差，因此泛化能力很强
  - Bagging方法的性能依赖于基学习器的稳定性。如果基学习器不稳定，Bagging有助于降低训练数据的随机波动导致的误差
  - 如果稳定，则集成学习器的误差主要由基学习器的偏倚引起，这时候采用Bagging方法可能不会在性能上有明显改善
  - 由于每个样本被选中的概率相同，因此Bagging并不侧重于训练数据集中的任何特定实例，因此对于噪声数据，Bagging方法一般不太受影响。

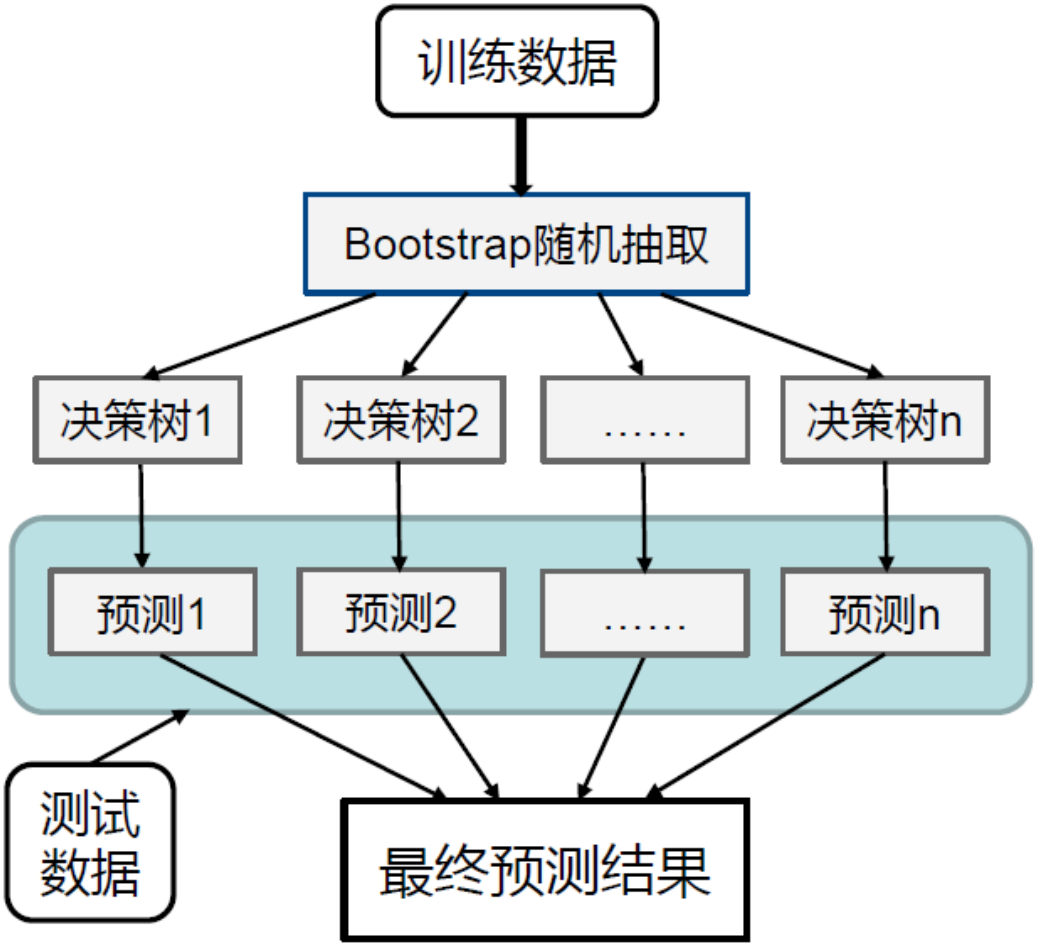
集成学习方法

- 随机森林
  - 是 Bagging 的扩展变体，它以决策树为基学习器构建Bagging集成，构成“森林”
  - 随机森林算法由很多决策树组成，每一棵决策树之间没有关联
  - 建立完森林后，当有新样本进入时，每棵决策树都会分别进行判断，然后基于投票法给出分类结果
  - 相较Bagging改进之处在于，进一步在决策树的训练过程中引入了随机特征选择

集成学习方法

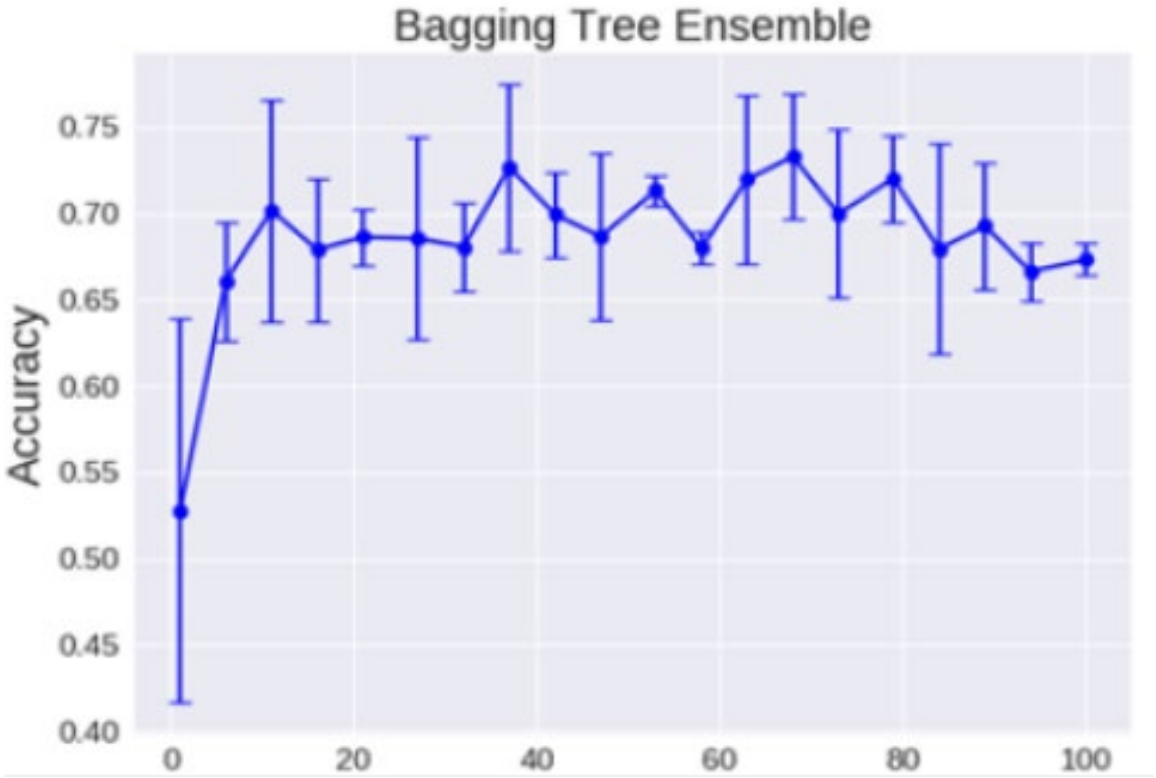
- 随机森林算法

1.从样本集中用Bootstrap采样选出 m个样本；
2.从所有属性中随机选择 K 个属性，选择出最佳分割属性作为节点创建决策树；
3.重复以上两步 n 次，即建立 n棵决策树；
4.这 n个决策树形成随机森林，通过投票表决结果决定数据属于那一类。



随机森林

- 注意:
  - 理论上构建越多的树效果会越好，但是实际上基本超过一定数量模型的准确率就差不多上下浮动了



随机森林

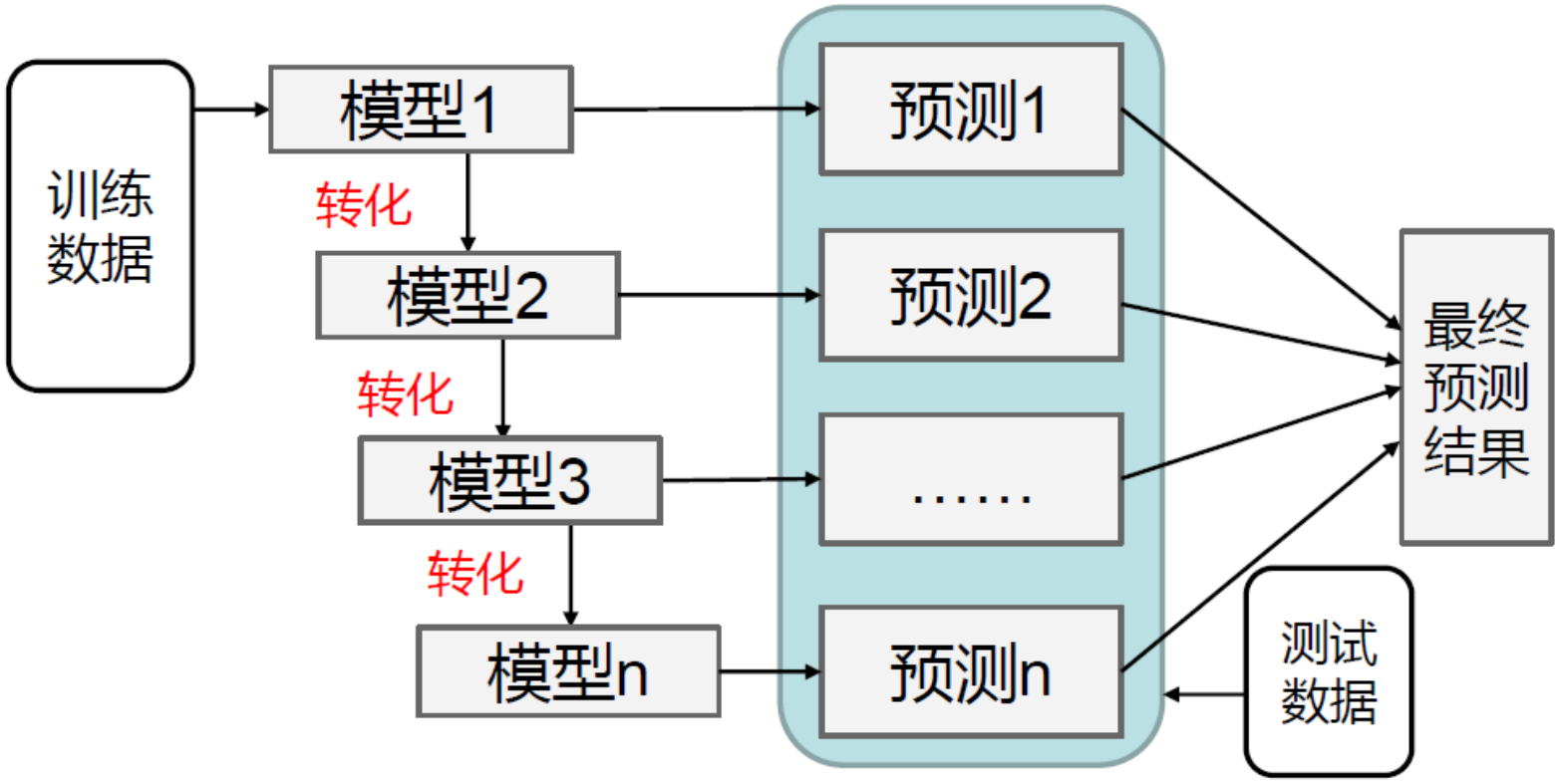
- 优点:
  - 在Bagging的基础上，进一步降低了模型的方差
  - 不需要给树做剪枝
  - 在大规模数据集，尤其是特征较多的情况下，表现较好
  - 可以处理高维数据，不用做特征选择，并且可以给出特征变量重要性的排序估计
- 缺点：
  - 随机森林已经被证明在某些噪音较大的分类或回归问题上会过拟合
  - 当随机森林中的决策树个数很多时，训练时需要的空间和时间会比较大
  - 可解释性一般

集成学习方法

- Boosting
  - 提升方法
    - 本质上是一种迭代方法
    - 训练过程为阶梯状，基模型按次序依次进行训练
    - 基模型的训练集按照某种策略每次都进行一定的转换，根据上一轮的结果动态调整每个样本的权重
    - 目的是通过后续模型尝试纠正先前模型的错误
    - 对所有基模型预测的结果进行线性综合产生最终的预测结果
    - 代表算法：Adaboost、GBDT、XGBoost、LightGBM

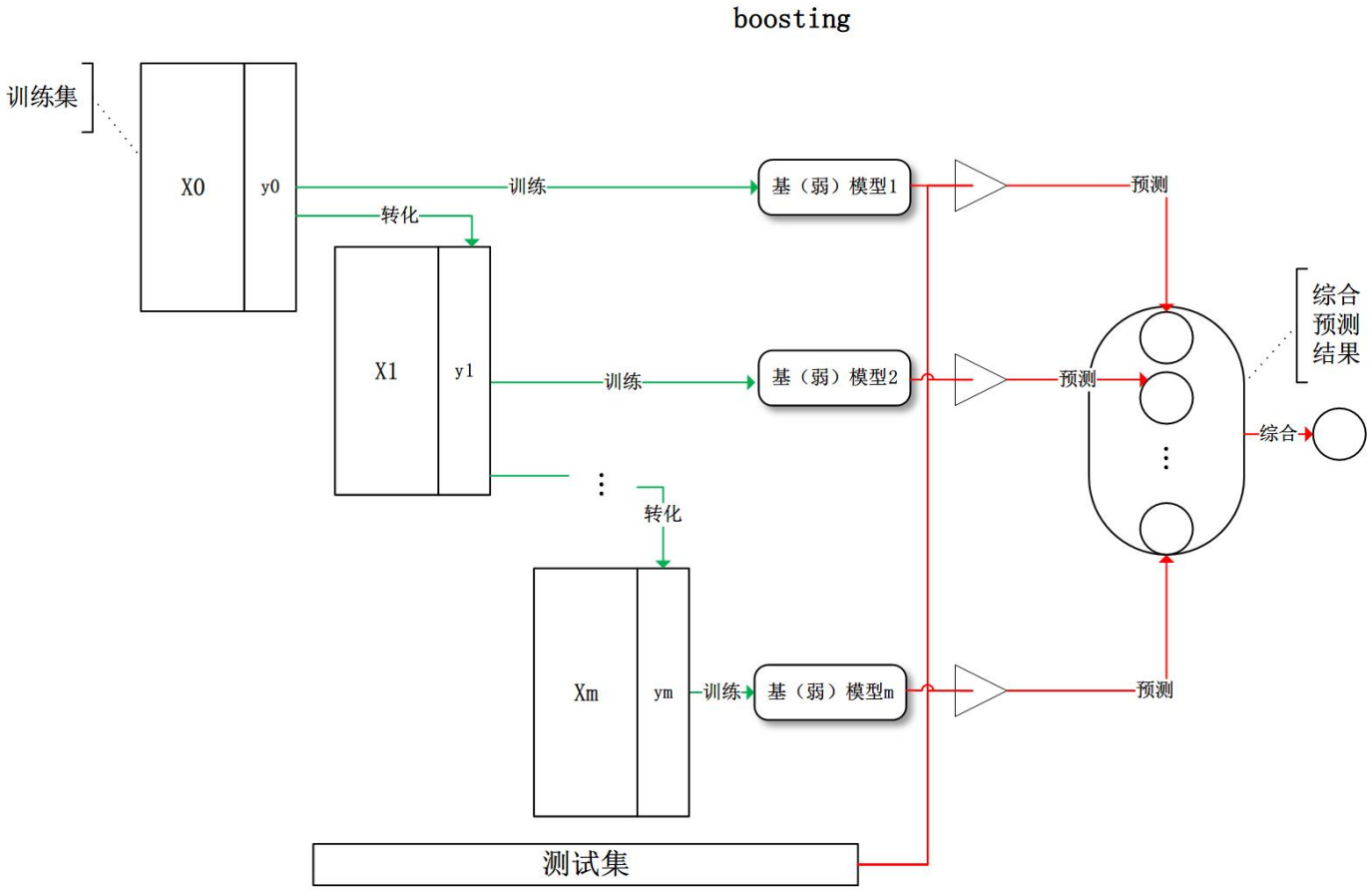
集成学习方法

- Boosting
  - 不同的训练集是通过调整每个样本对应的权重实现的，不同的权重对应不同的样本分布



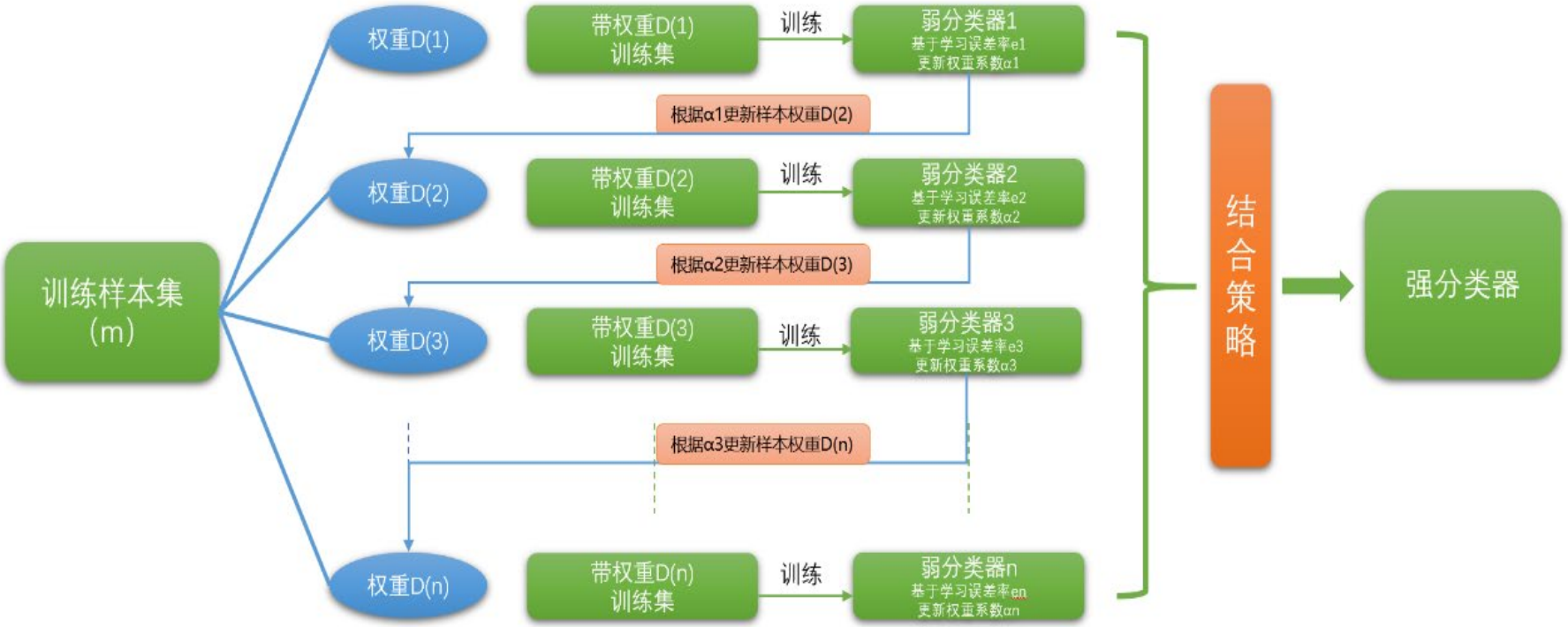
集成学习方法

- Boosting
  - 每次都根据上次的效果调整训练集，从而获得提升



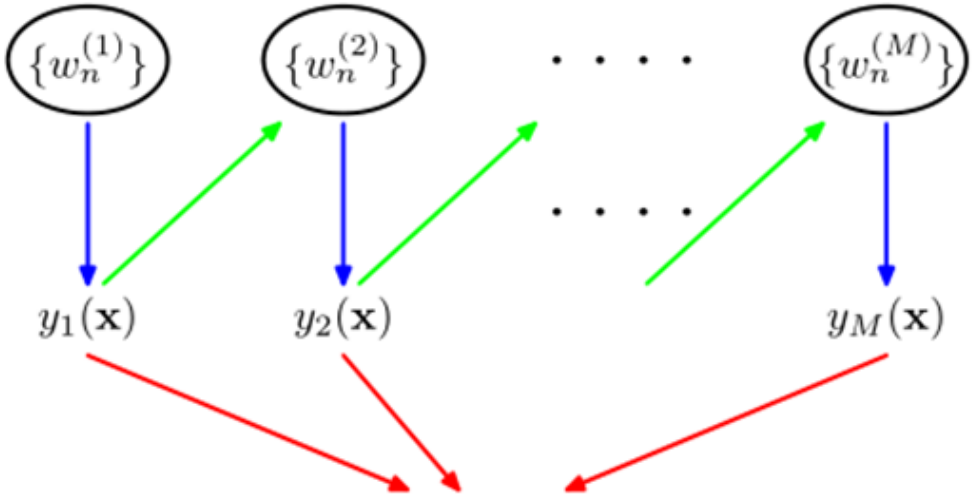
集成学习方法

- 每一轮如何改变训练数据的权值或概率分布？
  - 提高那些在前一轮分错样例的权值，减小前一轮分对样例的权值



集成学习方法

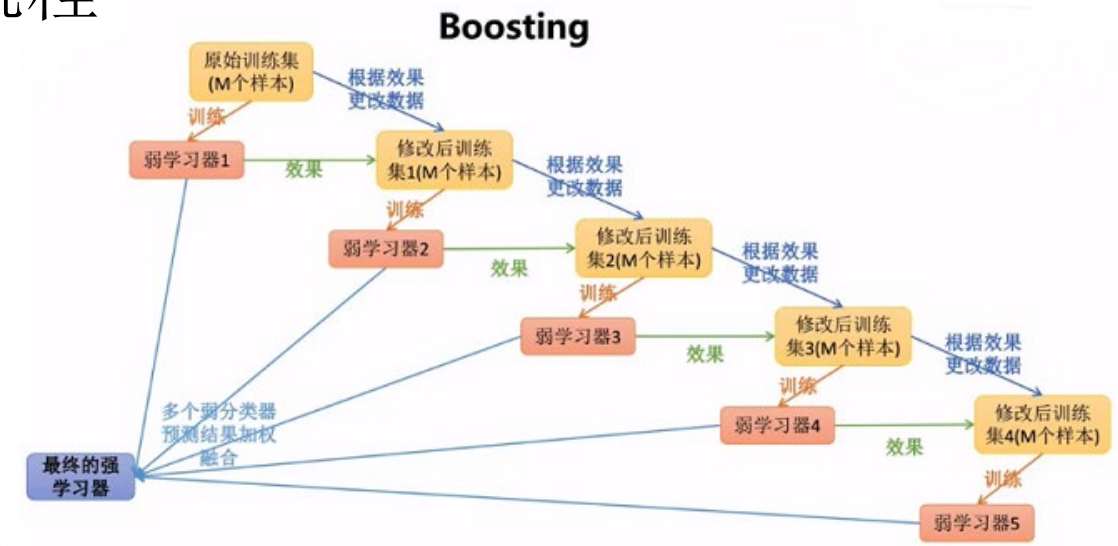
- Boosting的结合策略
  - 一般采用线性加权表决的方法（adaboost）
  - 加大误差率小的基学习器的权值，使其在表决中起较大的作用；减小误差率大的基学习器的权值，使其在表决中起较小的作用



$$Y_M(\mathbf{x}) = \text{sign}\left(\sum_m^M \alpha_m y_m(\mathbf{x})\right)$$

集成学习方法

- Boosting算法基本流程



Input: Sample distribution $\mathcal{D}$;
Base learning algorithm $\mathfrak{L}$;
Number of learning rounds T.

Process:

1. $\mathcal{D}_1 = \mathcal{D}$. % Initialize distribution
2. **for** $t = 1, \ldots, T$:
3. $h_t = \mathfrak{L}(\mathcal{D}_t)$; % Train a weak learner from distribution $\mathcal{D}_t$
4. $\epsilon_t = P_{\boldsymbol{x} \sim D_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$; % Evaluate the error of h_t
5. $\mathcal{D}_{t+1} = Adjust\_Distribution(\mathcal{D}_t, \epsilon_t)$
6. **end**

Output: $H(\boldsymbol{x}) = Combine\_Outputs(\{h_1(\boldsymbol{x}), \ldots, h_t(\boldsymbol{x})\})$

集成学习方法

- Review
  - 什么是集成学习？核心思想？
  - 获得不同基学习器的方法？
  - Bagging的特点？
  - Boosting的特点？
  - 这两种方法的适用性和优缺点？

集成学习方法

- AdaBoost（Adaptive Boosting，自适应增强）
  - 1995， Freund和Schapire提出
  - 其自适应在于：前一个基本分类器分错的样本会得到加强，加权后的全体样本再次被用来训练下一个基本分类器
  - 在每一轮中加入一个新的弱分类器，直到达到某个预定的足够小的错误率或达到预先指定的最大迭代次数。
  - 后一个模型的训练永远是在前一个模型的基础上完成！

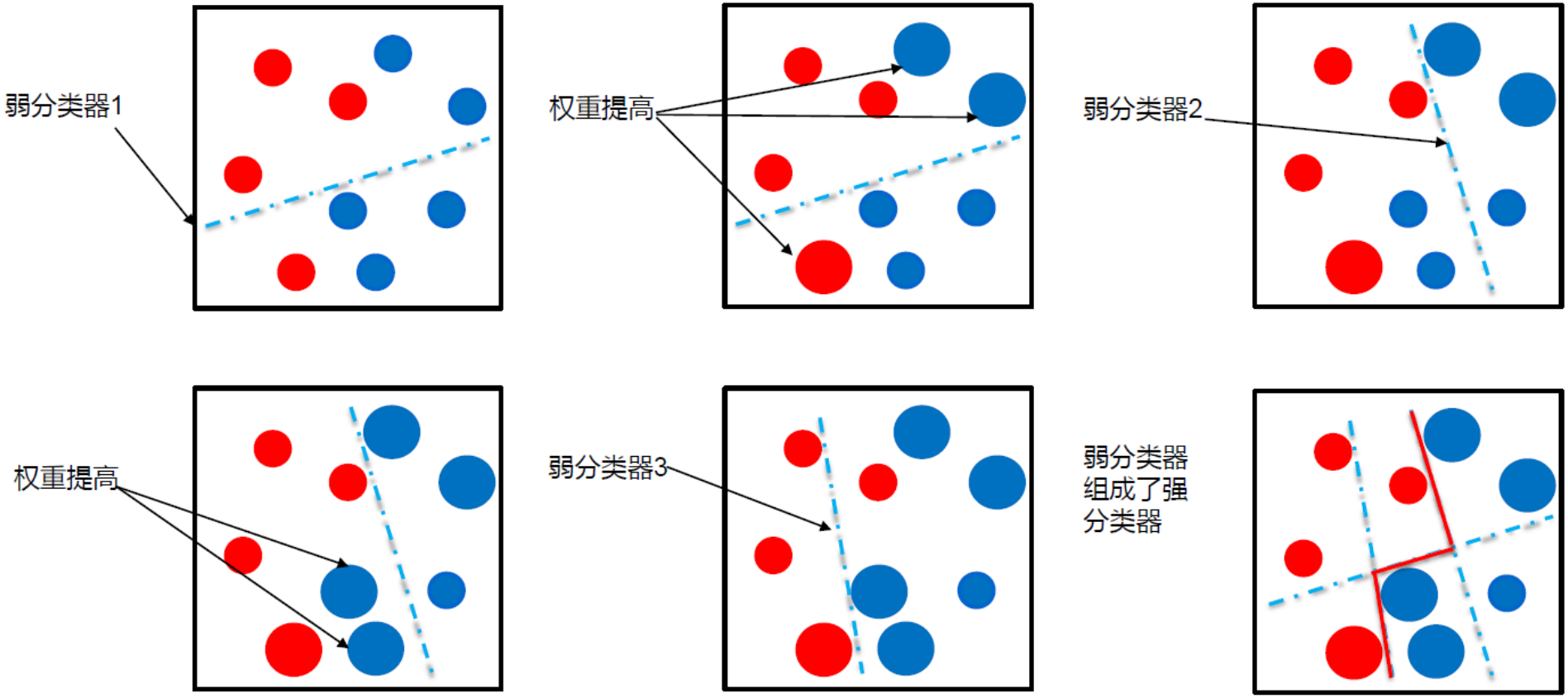
集成学习方法

- AdaBoost算法思想

1. 首先赋予每个训练样本相同的初始化权重，在此训练样本分布下训练出一个弱学习器；
2. 利用该弱学习器更新每个样本的权重（由分类/预测误差决定），分类错误(预测误差大)的样本认为是困难样本，权重增加，反之权重降低，得到一个新的样本分布；
3. 在新的样本分布下，在训练一个新的弱学习器，并且更新样本权重，重复以上过程T次，得到T个弱学习器
4. 加权集成这些弱学习器得到一个强学习器(权重同样由每个学习器的误差决定，误差低的学习器权重大)

集成学习方法

- AdaBoost算法思想



集成学习方法

- AdaBoost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: $\mathcal{D}_1(\boldsymbol{x}) = 1/m$.
2: **for** $t = 1, 2, \ldots, T$ **do**
3: $h_t = \mathfrak{L}(D, \mathcal{D}_t)$;
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: $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$;
7: $\mathcal{D}_{t+1}(\boldsymbol{x}) = \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t} \times \begin{cases} \exp(-\alpha_t), & \text{if } h_t(\boldsymbol{x}) = f(\boldsymbol{x}) \\ \exp(\alpha_t), & \text{if } h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}) \end{cases}$
$= \frac{\mathcal{D}_t(\boldsymbol{x}) \exp(-\alpha_t f(\boldsymbol{x}) h_t(\boldsymbol{x}))}{Z_t}$
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

集成学习方法

- AdaBoost算法流程

$I(G_m(x_i) \neq y_i)$:

当$G_m(x_i)$与y_i相等时，函数取值为0

当$G_m(x_i)$与y_i不相等时，取值为1

$G_m(x)$在训练数据集上的误差率e_m

就是被$G_m(x)$误分类样本的权值之和

当$e_k \leq 0.5$时，$\alpha_k \geq 0$

α_k随着e_k的减小而增大

输入：训练数据集$D = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(M)}, y^{(M)})\}$，$x^{(i)} \in \mathcal{X} \subseteq R^N$，$y^{(i)} \in \mathcal{Y} = \{-1, +1\}$，弱分类器；

输出：最终分类器$G(x)$.

过程：

(1). 初始化训练数据的权值分布

$$D_1 = (w_{11}, w_{12}, \cdots, w_{1M}), \quad w_{1i} = \frac{1}{M}, \ i = 1, 2, \cdots, M$$

(2). 训练K个弱分类器$k = 1, 2, \cdots, K$

(a). 使用具有权值分布D_k的训练数据集学习，得到基本分类器

$$G_k(x) : \mathcal{X} \to \{-1, +1\} \qquad (ml.1.6.3)$$

(b). 计算$G_k(x)$在训练数据集上的分类误差率

$$e_k = P(G_k(x^{(i)}) \neq y^{(i)}) = \sum_{i=1}^{M} w_{ki} I(G_k(x^{(i)}) \neq y^{(i)}) \qquad (ml.1.6.4)$$

(c). 计算$G_k(x)$的系数

$$\alpha_k = \frac{1}{2} \log \frac{1 - e_k}{e_k} \quad (e\text{是自然对数}) \qquad (ml.1.6.5)$$

(d). 更新训练数据集的权值分布

$$D_{k+1} = (w_{k+1,1}, w_{k+1,2}, \cdots, w_{k+1,M})$$

$$w_{k+1,i} = \frac{w_{k,i}}{Z_k} \exp(-\alpha_k y^{(i)} G_k(x^{(i)})), \quad i = 1, 2, \cdots, M \qquad (ml.1.6.6)$$

Z_k是规范化因子

$$Z_k = \sum_{i=1}^{M} w_{k,i} \cdot \exp(-\alpha_k y^{(i)} G_k(x^{(i)})) \qquad (ml.1.6.7)$$

使D_{k+1}成为一个概率分布。

(3). 构建基本分类器的线性组合

$$f(x) = \sum_{k=1}^{K} \alpha_k G_k(x) \qquad (ml.1.6.8)$$

得到最终的分类器

$$G(x) = sign(f(x)) = sign\left(\sum_{k=1}^{K} \alpha_k G_k(x)\right) \qquad (ml.1.6.9)$$

集成学习方法

- AdaBoost案例

  假定有如下数据

序号	i	1	2	3	4	5	6	7	8	9	10
数据	x	0	1	2	3	4	5	6	7	8	9
类别标签	y	1	1	1	-1	-1	-1	1	1	1	-1

初始化：$w_{1i} = \frac{1}{n} = 0.1$，n=10（样本个数）

序号	i	1	2	3	4	5	6	7	8	9	10
数据	x	0	1	2	3	4	5	6	7	8	9
类别标签	y	1	1	1	-1	-1	-1	1	1	1	-1
初始权值	w_{1i}	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

阈值猜测：观察数据，可以发现数据分为两类：-1和1，其中数据“0,1,2”对应“1”类，数据“3,4,5”对应“-1”类，数据“6,7,8”对应“1”类，数据“9”对应“"1”类。抛开单个的数据“9”，可以找到对应的数据分界点（即可能的阈值），比如：2.5、5.5、8.5（一般0.5的往上加，也可以是其他数）。然后计算每个点的误差率，选最小的那个作为阈值。

集成学习方法

- AdaBoost案例

计算可知，阈值取2.5 或8.5时，误差率一样，所以可以任选一个作为基本分类器。这里选2.5为例。

$$G_1(x) = \begin{cases} 1, & x < 2.5 \\ -1, & x > 2.5 \end{cases}$$

序号	i	1	2	3	4	5	6	7	8	9	10
数据	x	0	1	2	3	4	5	6	7	8	9
类别标签	y	1	1	1	-1	-1	-1	1	1	1	-1
分类器结果	$G_1(x)$	1	1	1	-1	-1	-1	-1	-1	-1	-1
分类结果		对	对	对	对	对	对	错	错	错	对

从上可得 $G_1(x)$ 在训练数据集上的误差率（被分错类的样本的权值之和）：

$$e_1 = P(G_1(x_i) \neq y_i) = \sum_{G_1(x_i) \neq y_i} w_{1i} = 0.1 + 0.1 + 0.1 = 0.3$$

2> 计算 $G_1(x)$ 的系数:

$$\alpha_1 = \frac{1}{2} ln \frac{1-e_1}{e_1} = \frac{1}{2} ln \frac{1-0.3}{0.3} \approx 0.42365$$

这个α_1 代表 $G_1(x)$ 在最终的分类函数中所占的比重约为0.42365,

集成学习方法

- AdaBoost案例

3> 分类函数

$$f_1(x) = \alpha_1 G_1(x) = 0.42365 G_1(x)$$

4> 更新权值分布：

根据公式：

$$w_{m+1,i} = \frac{w_{mi}}{Z_m} e^{-\alpha_m y_i G_m(x_i)}, \quad i = 1, 2, \cdots, N$$

$$Z_m = \sum_{i=1}^{N} w_{mi}\ e^{-\alpha_m y_i G_m(x_i)}$$

序号	i	1	2	3	4	5	6	7	8	9	10
数据	x	0	1	2	3	4	5	6	7	8	9
类别标签	y	1	1	1	-1	-1	-1	1	1	1	-1
初始权值 1	w_{1i}	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
更新权值 2	w_{2i}	0.07143	0.07143	0.07143	0.07143	0.07143	0.07143	0.16666	0.16666	0.16666	0.07143

集成学习方法

- AdaBoost案例

迭代过程2：m=2

1> 确定阈值的取值及误差率

由上面可知，阈值取8.5时，误差率最小，所以：

$$G_2(x) = \begin{cases} 1, & x < 8.5 \\ -1, & x > 8.5 \end{cases}$$

计算误差率：

序号	i	1	2	3	4	5	6	7	8	9	10
数据	x	0	1	2	3	4	5	6	7	8	9
类别标签	y	1	1	1	-1	-1	-1	1	1	1	-1
权值分布	w_{2i}	0.07143	0.07143	0.07143	0.07143	0.07143	0.07143	0.16666	0.16666	0.16666	0.07143
分类器结果	$G_2(x)$	1	1	1	1	1	1	1	1	1	-1
分类结果		对	对	对	错	错	错	对	对	对	对

$$e_2 = P(G_2(x_i) \neq y_i) = \sum_{G_2(x_i) \neq y_i} w_{2i} = 0.07143 + 0.07143 + 0.07143 = 0.21429$$

集成学习方法

- AdaBoost案例

2> 计算 $G_2(x)$ 的系数:

$$\alpha_2 = \frac{1}{2} ln \frac{1-e_2}{e_2} = \frac{1}{2} ln \frac{1-0.21429}{0.21429} \approx 0.64963$$

3> 分类函数

$$f_2(x) = \alpha_2 G_2(x) = 0.64963 G_2(x)$$

4> 更新权值分布:

序号	i	1	2	3	4	5	6	7	8	9	10
数据	x	0	1	2	3	4	5	6	7	8	9
类别标签	y	1	1	1	-1	-1	-1	1	1	1	-1
初始权值 1	w_{1i}	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
权值 2	w_{2i}	0.07143	0.07143	0.07143	0.07143	0.07143	0.07143	0.16666	0.16666	0.16666	0.07143
更新权值 3	w_{3i}	0.04546	0.04546	0.04546	0.16667	0.16667	0.16667	0.10606	0.10606	0.10606	0.04546

集成学习方法

- AdaBoost案例

如此迭代下去，第三次迭代后的误差率为0，所以迭代可以到此结束。

序号	i	1	2	3	4	5	6	7	8	9	10
数据	x	0	1	2	3	4	5	6	7	8	9
类别标签	y	1	1	1	-1	-1	-1	1	1	1	-1
初始权值1	w_{1i}	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
权值2	w_{2i}	0.07143	0.07143	0.07143	0.07143	0.07143	0.07143	0.16666	0.16666	0.16666	0.07143
权值3	w_{3i}	0.04546	0.04546	0.04546	0.16667	0.16667	0.16667	0.10606	0.10606	0.10606	0.04546
更新权值4	w_{4i}	0.12500	0.12500	0.12500	0.10185	0.10185	0.10185	0.06481	0.06481	0.06481	0.12500

最终分类器：

$$G_m(x) = sign(0.42365G_1(x) + 0.64963G_2(x)0.75197G_3(x))$$

Boosting总结

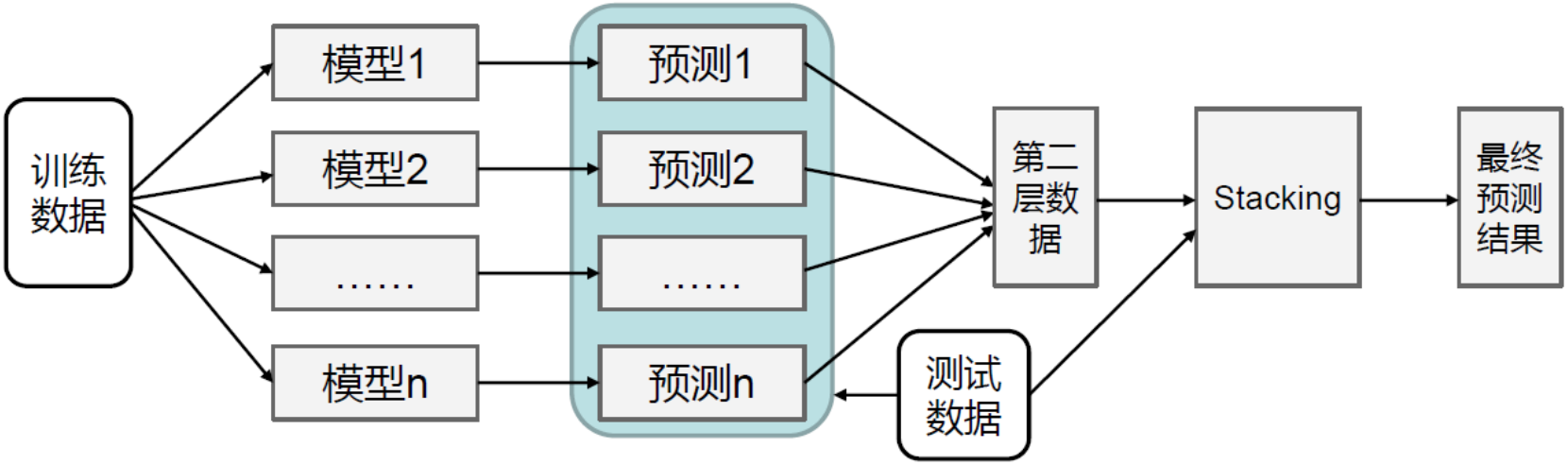
- 优点:
  - 很好的利用了弱学习器进行级联
  - 不断重点学习分类错误样本，从而降低了预测的偏差，具有很高的精度
  - 相对于Bagging算法和Random Forest算法，AdaBoost充分考虑的每个分类器的权重
- 缺点：
  - AdaBoost迭代次数也就是弱学习器数目不太好设定，可以使用交叉验证来进行确定。
  - 训练比较耗时，串行且每次重新选择分类器最好的切分点
  - 对训练样本的噪声非常敏感，主要任务集中在噪声样本上，从而影响最终的分类性能
  - 噪声过大时，也会产生过拟合

Bagging和Boosting对比

- 差异:
  - 样本选择：Bagging算法是有放回的随机采样；Boosting算法是每一轮训练集不变，只是训练集中的每个样例在分类器中的权重发生变化，而权重根据上一轮的分类结果进行调整
  - 样例权重：Bagging使用随机抽样，样例的权重想通;Boosting根据错误率不断的调整样例的权重值，错误率越大则权重越大
  - 预测函数：Bagging所有预测模型的权重相等;Boosting算法对于误差小的分类器具有更大的权重
  - 并行计算：Bagging算法可以并行生成各个基模型;Boosting理论上只能顺序生产，因为后一个模型需要前一个模型的结果
  - Bagging是减少模型的variance(方差)，对解决过拟合有效；
  - Boosting是减少模型的Bias(偏差)，对解决欠拟合有效。

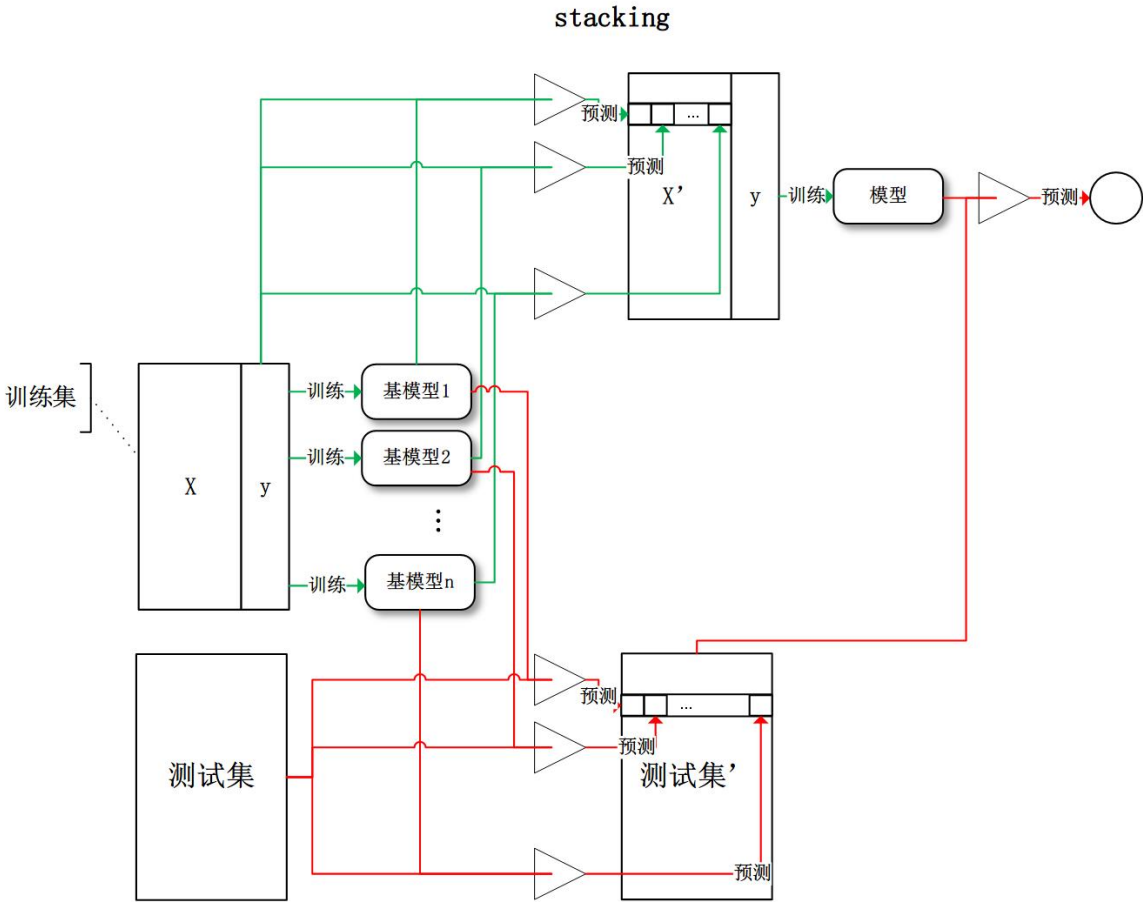
Stacking

- 堆叠:
  - Stacking是指训练一个模型用于组合(combine)其它模型(基模型/基学习器)的技术。即首先训练出多个不同的模型，然后再以之前训练的各个模型的输出作为输入来新训练一个新的模型，从而得到一个最终的模型。
  - 第j 个基模型对第i 个训练样本的预测值将作为新的训练集中第i 个样本的第j 个特征值，最后基于新的训练集进行训练。同理，预测的过程也要先经过所有基模型的预测形成新的测试集，最后再对测试集进行预测



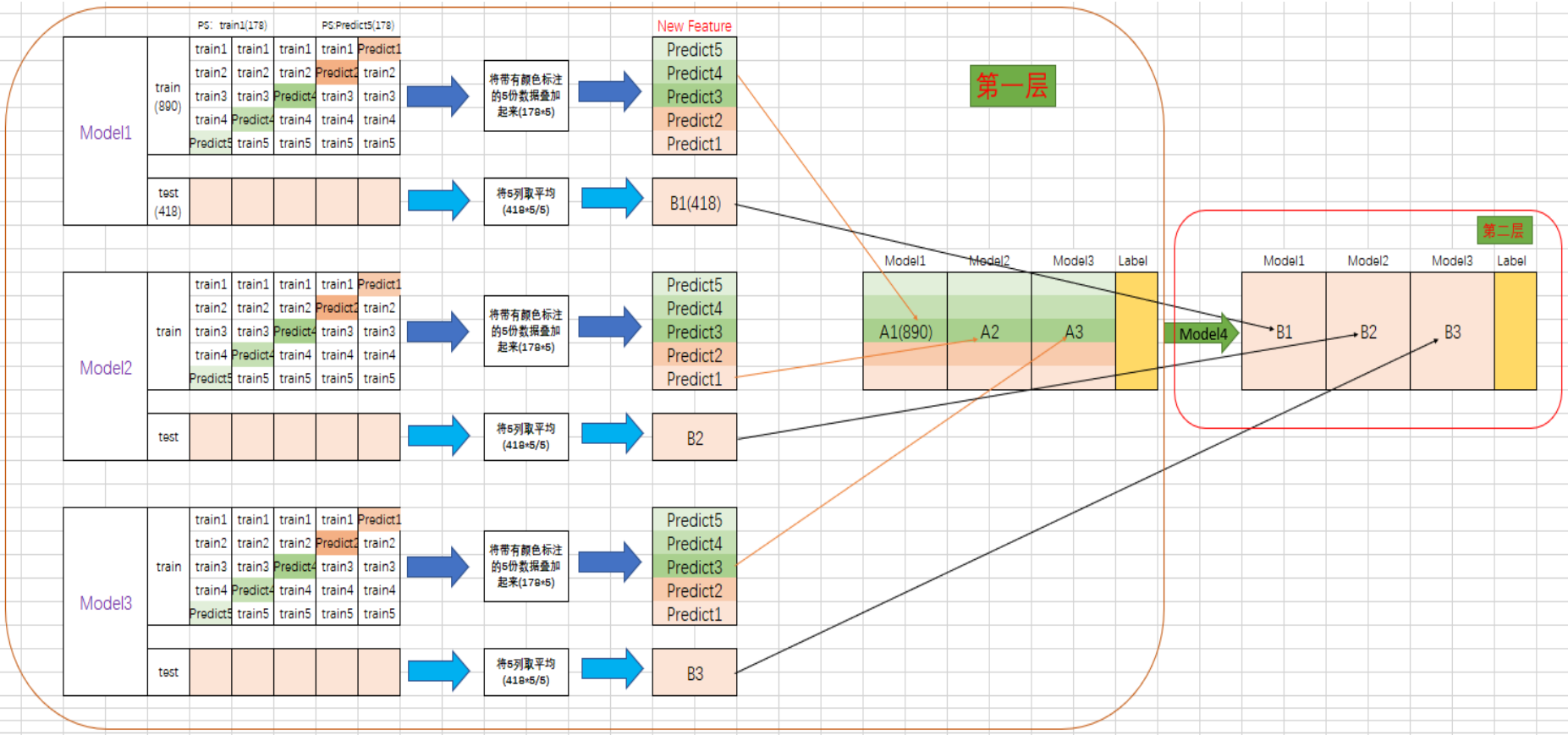
Stacking

- 堆叠:
  - 在Stacking中，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学习器或者元学习器。
  - Stacking可以看作是Bagging和Boosting的一种结合
  - Stacking一般分为两层，第一层为多个初级学习器，第二层为单个次级学习器



Stacking

- 堆叠:
  - Stacking一般分为两层，第一层为多个初级学习器，第二层为单个次级学习器



Stacking

- 堆叠:

1、首先划分数据：训练集 train 和测试集 test。将训练集 train 分成5份：train1，train2，train3，train4，train5

2、选定基模型Model1，依次用 train1，train2，train3，train4，train5作为验证集，其余4份作为训练集，进行5折交叉验证进行模型训练

3、将每份验证集上的输出按行拼接起来作为新的特征A1, 训练好的Model1在测试集 test 上进行预测，得到预测值向量B1。

4、假设第一层有3个模型，同样的方式训练一遍， 可以得到 A2、A3 和 B2、B3 。

5、这样你就会得到：来自 5-fold 的预测值(A1，A2，A3） 和 来自 test 的预测值（B1, B2, B3）。

6、来自5-fold的预测值矩阵 （A1，A2，A3）作为新的train Data，训练第二层的模型(Model4)。

7、来自 test Data 预测值矩阵 （B1, B2, B3） 就是新的test Data，用训练好的模型来预测。

Stacking

- 算法流程:

输入: 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
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:

```
for t = 1, 2, ..., T do
    h_t = L_t(D);
end for
D' = ∅;
for i = 1, 2, ..., m do
    for t = 1, 2, ..., T do
        z_it = h_t(x_i);
    end for
    D' = D' ∪ ((z_i1, z_i2, ..., z_iT), y_i);
end for
h' = L(D');
```

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

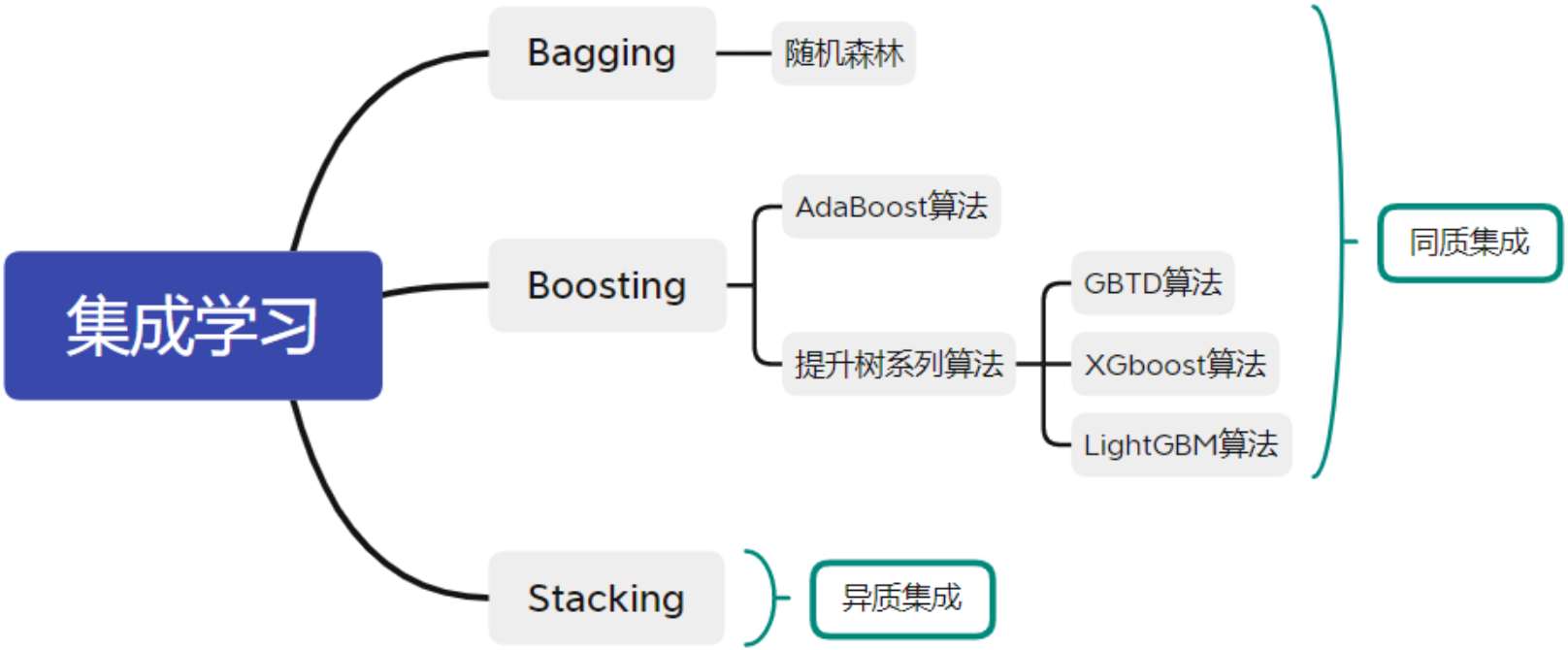
图 8.9 Stacking 算法

Stacking

- 注意事项:
  - 第一层的基模型最好是强模型，而第二层的基模型可以放一个简单的分类器，防止过拟合，一般采用线性方法，如逻辑回归(LR)、MLR等
  - 第一层基模型的个数不能太小，因为一层模型个数等于第二层分类器的特征维度
  - 第一层的基模型须“准而不同”，如果有一个性能很差的模型出现在第一层，将会影响整个模型整合后的效果
  - Stacking的模型整合方法也称作学习法，即将训练集弱学习器的学习结果作为输入，重新训练一个学习器来得到最终结果
  - 优点: 性能好、准确率高
  - 缺点：数据量要求较大、速度慢、过拟合风险

集成学习的分类

- Boosting和Bagging通常都是使用同一种基学习器，因此我们一般也称之为同质集成方法。Stacking通常都是基于多个不同的基学习器做的集成，因此我们也称之为异质集成方法。



总结

- 集成学习：
  - 将多个算法聚集在一起，以提高分类/预测的准确率。
  - 这些算法可以是不同的算法，也可以是相同的算法
  - 集成学习由训练数据构建一组基学习器，然后通过对每个基学习器的预测进行投票、平均或学习来进行整合
  - 严格来说，集成学习并不算是一种算法，而是一种模型的整合方法。
  - 通常集成后的性能会好于单个模型
  - 进行集成学习是为了达到达到减小方差（bagging）、偏差（boosting）或改进预测（stacking）的目的