集成学习
Ensemble Learning

集成学习

我们之前讨论的学习器都是单一的，独立的。

整体表现比较差的学习器，在一些样本上的表现是否有可能会超过“最好”的学习器。

集成学习

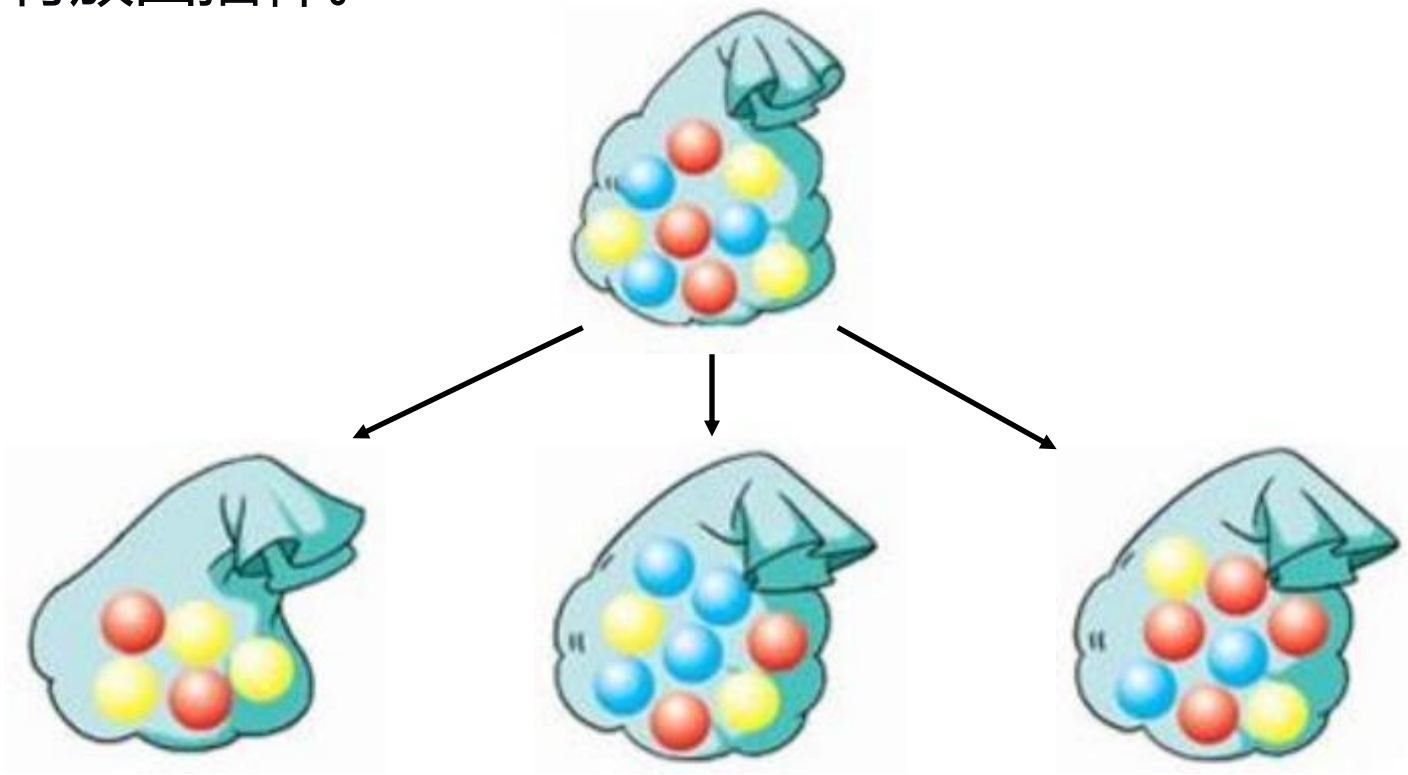
当做重要决定时，大家可能都会考虑吸取多个专家而不只是一个人的意见。集成学习也是如此。

集成学习就是组合多个学习器，最后可以得到一个更好的学习器。

集成学习算法：
1.个体学习器之间不存在强依赖关系，装袋（bagging）
2.随机森林（Random Forest）
3.个体学习器之间存在强依赖关系，提升（boosting）
4.Stacking

bagging

直觉：数据量越大，学习器性能越好。
bagging也叫做bootstrap aggregating，是在原始数据集选择S次后得到S个新数据集的一种技术。是一种有放回抽样。



bagging

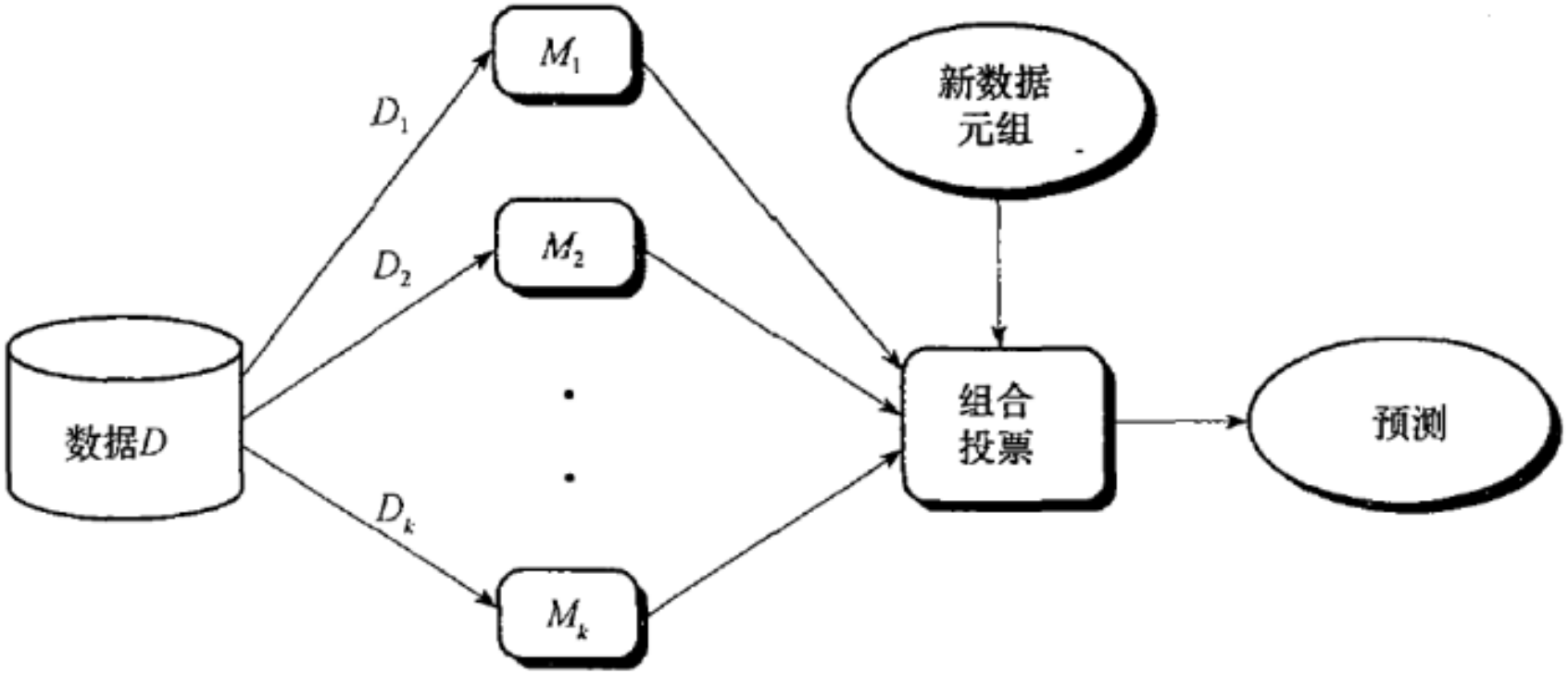
原始训练数据集

{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

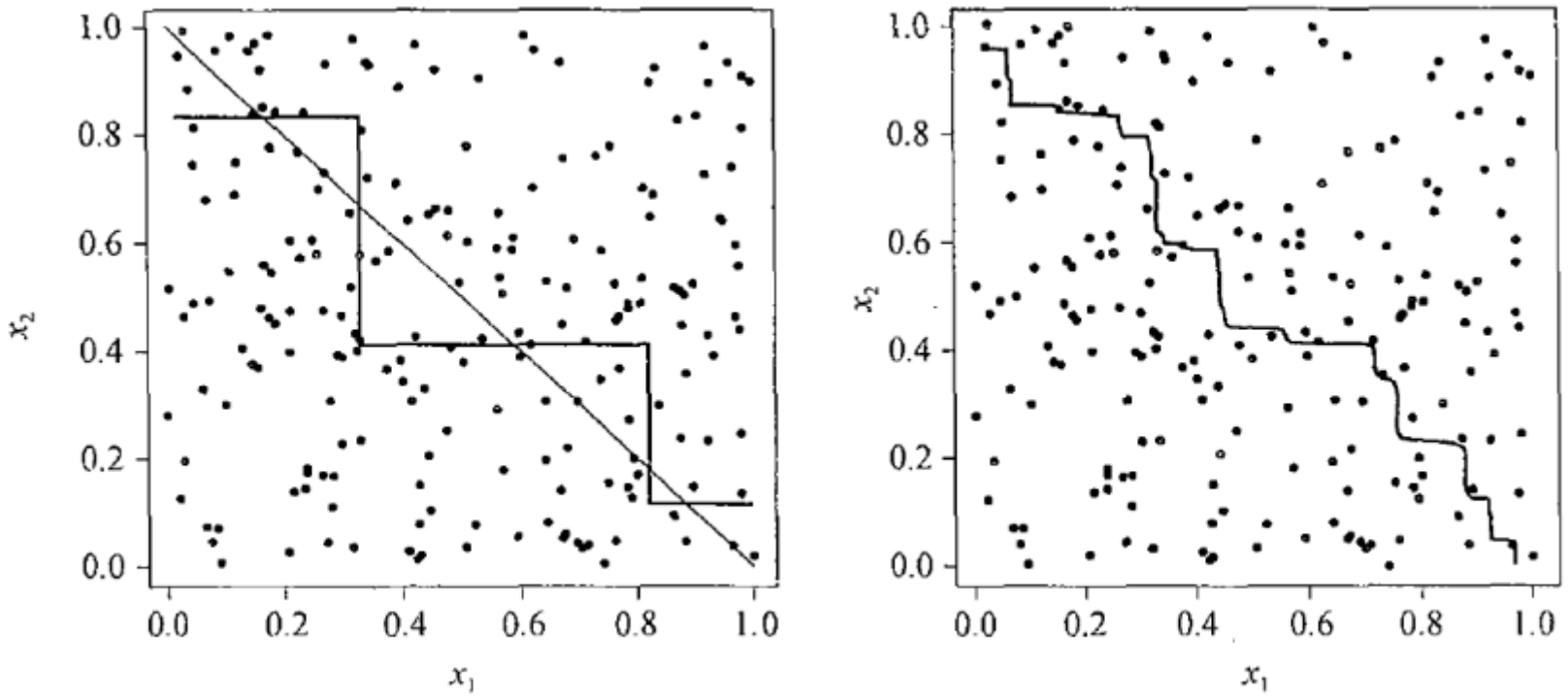
Bootstrap采样

{7, 2, 6, 7, 5, 4, 8, 8, 1, 0} —— 未采样3, 9
{1, 3, 8, 0, 3, 5, 8, 0, 1, 9} —— 未采样2, 4, 6, 7
{2, 9, 4, 2, 7, 9, 3, 0, 1, 0} —— 未采样3, 5, 6, 8

bagging



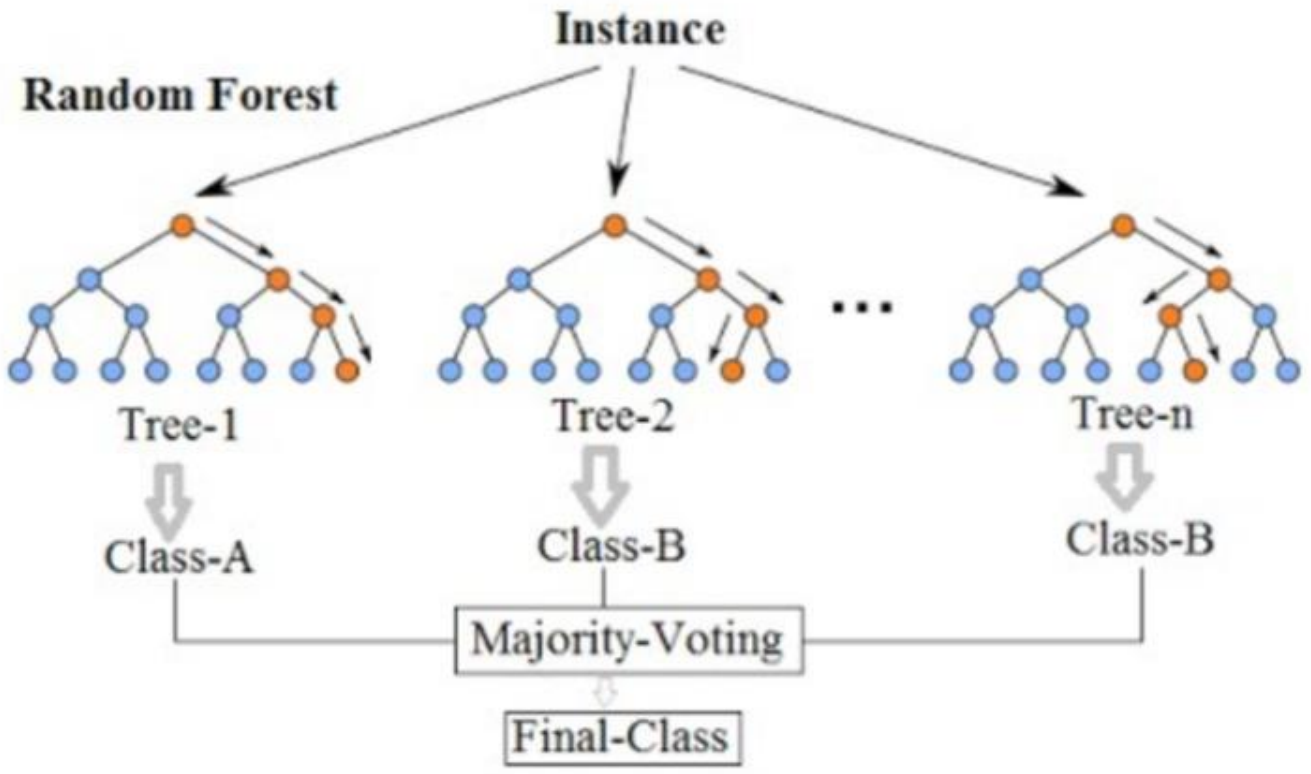
bagging


1.0
0.8
0.6
0.4
0.2
0.0
x_2
0.0
0.2
0.4
0.6
0.8
1.0
x_1
1.0
0.8
0.6
0.4
0.2
0.0
x_2
0.0
0.2
0.4
0.6
0.8
1.0
x_1

Bagging

随机森林(Random Forest)

RF = 决策树+Bagging+随机属性选择



RF算法流程

1.样本的随机：从样本集中用bagging的方式，随机选择n个样本。

2.特征的随机：从所有属性d中随机选择k个属性(k<d)，然后从k个属性中选择最佳分割属性作为节点建立CART决策树。

3.重复以上两个步骤m次，建立m棵CART决策树。

4.这m棵CART决策树形成随机森林，通过投票表决结果，决定数据属于哪一类。

随机森林

boosting

AdaBoost是英文“Adaptive Boosting”（自适应增强）的缩写，它的自适应在于：前一个基本分类器被错误分类的样本的权值会增大，而正确分类的样本的权值会减小，并再次用来训练下一个基本分类器。同时，在每一轮迭代中，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才确定最终的强分类器。

boosting

直觉：将学习器的重点放在“容易”出错的样本上。可以提升学习器的性能。

原始训练数据集

0	1	2	3	4	5	6	7	8	9

Boosting采样

Iteration 1:	7	2	6	7	5	4	8	8	1	0
Iteration 2:	1	3	8	4	3	5	4	0	1	4
Iteration 3:	4	9	4	2	4	4	3	0	1	4

boosting

Adaboost算法可以简述为三个步骤：
（1）首先，是初始化训练数据的权值分布D_1。假设有N个训练样本数据，则每一个训练样本最开始时，都被赋予相同的权值：$w_1=1/N$。
（2）然后，训练弱分类器h_i。具体训练过程中是：如果某个训练样本点，被弱分类器h_i准确地分类，那么在构造下一个训练集中，它对应的权值要减小；相反，如果某个训练样本点被错误分类，那么它的权值就应该增大。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。

boosting

（3）最后，将各个训练得到的弱分类器组合成一个强分类器。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。

换而言之，误差率低的弱分类器在最终分类器中占的权重较大，否则较小。

Adaboost训练过程

算法： Adaboost.一种提升算法——创建分类器的组合。每个给出一个加权投票。

输入：

- D：类标记的训练元组集。
- k：轮数（每轮产生一个分类器）。
- 一种分类学习方案。

输出： 一个复合模型。

方法：

(1) 将D中每个元组的权重初始化为$1/d$;
(2) **for** i = 1 **to** k **do** // 对于每一轮
(3) 根据元组的权重从D中有放回抽样,得到D_i;
(4) 使用训练集D_i导出模型M_i;
(5) 计算M_i的错误率$error(M_i)$（8.34式）
(6) **if** $error(M_i) > 0.5$ **then**
(7) 转步骤（3）重试;
(8) **endif**
(9) **for** D_i的每个被正确分类的元组**do**
(10) 元组的权重乘以$error(M_i)/(1-error(M_i))$; // 更新权重
(11) 规范化每个元组的权重;
(12) **endfor**

Adaboost判断过程

使用组合分类器对元组x分类：

(1) 将每个类的权重初始化为0；

(2) **for** $i = 1$ to k **do** // 对于每个分类器

(3) $w_i = \log \frac{1-error(M_i)}{error(M_i)}$； // 分类器的投票权重

(4) $c = M_i(\mathbf{x})$； // 从M_i得到$\mathbf{x}$的类预测

(5) 将w_i加到类c的权重；

(6) **endfor**

(7) 返回具有最大权重的类；

Adaboost算法流程2

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 $\mathfrak{L}$; 可以是各种机器学习算法
训练轮数 T. T轮可以得到T个基学习器

过程：

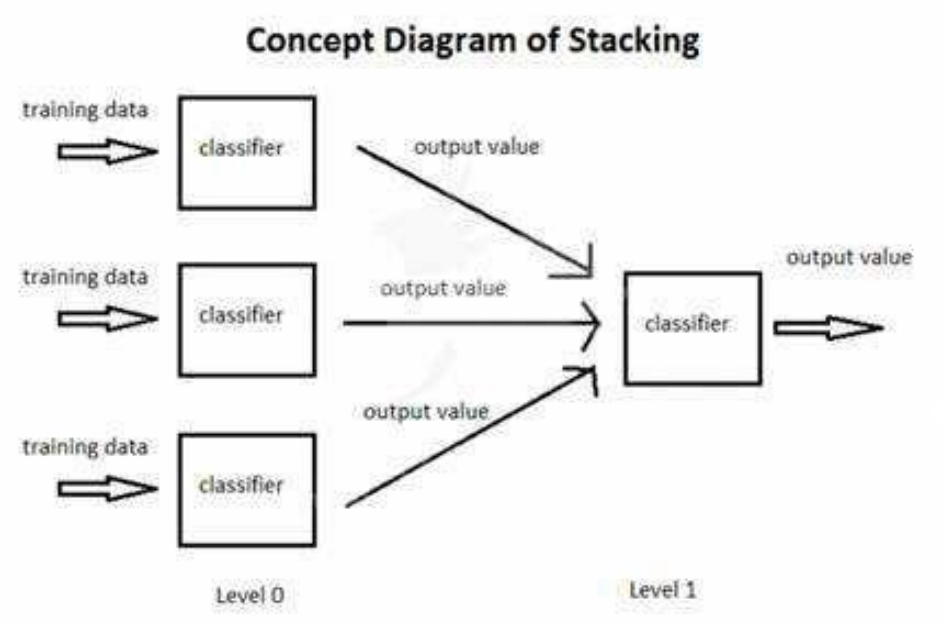
1: $\mathcal{D}_1(\boldsymbol{x}) = 1/m.$ 训练集刚开始是均匀分布，初始化权值都是1/m
2: **for** $t = 1, 2, \ldots, T$ **do**
3: $h_t = \mathfrak{L}(D, \mathcal{D}_t)$; 机器学习算法在D训练集基于$D_t$分布下训练得到$h_t$基学习器
4: $\epsilon_t = P_{\boldsymbol{x} \sim \mathcal{D}_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$;计算$h_t$基学习器在训练集$D_t$分布下的错误率$\epsilon_t$
5: **if** $\epsilon_t > 0.5$ **then break**
6: $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$; 由错误率得到一个权重$\alpha_t$ ， 错误率越低，权值越大，α_t>=0
7: $\mathcal{D}_{t+1}(\boldsymbol{x}) = \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t} \times \begin{cases} \exp(-\alpha_t), & \leq 1 \text{ if } h_t(\boldsymbol{x}) = f(\boldsymbol{x}) \text{ 分类正确，权重变小} \\ \exp(\alpha_t), & \geq 1 \text{ if } h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}) \text{ 分类错误，权重变大} \end{cases}$
$= \frac{\mathcal{D}_t(\boldsymbol{x}) \exp(-\alpha_t f(\boldsymbol{x}) h_t(\boldsymbol{x}))}{Z_t}$ Z_t归一化
8: **end for**

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

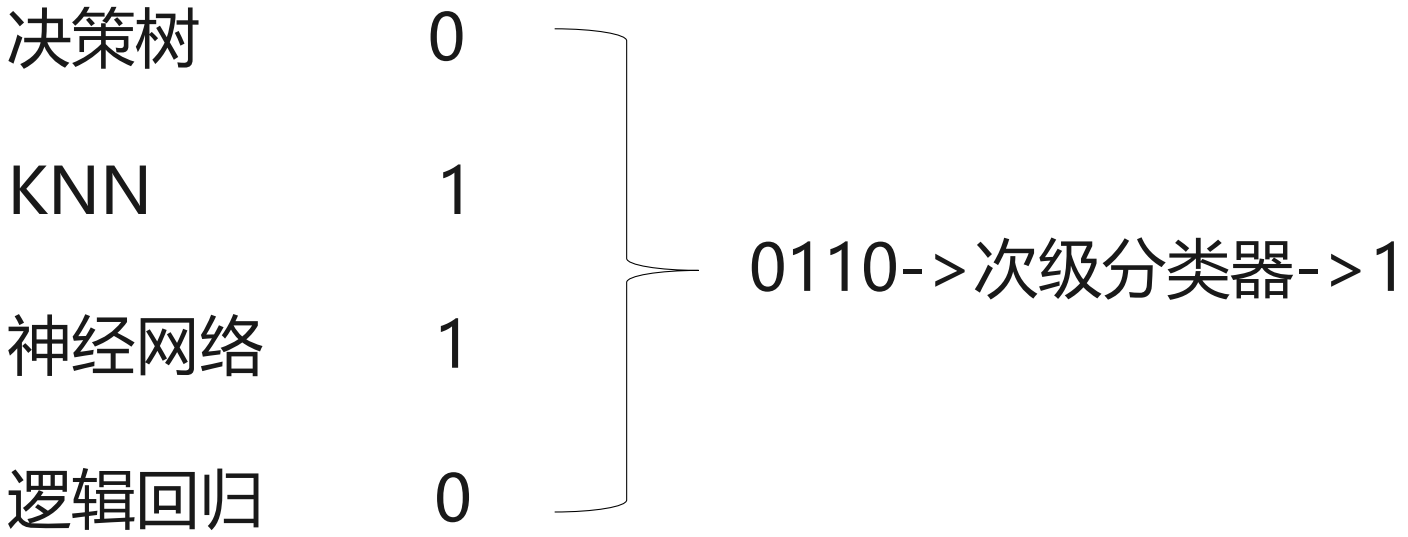
Adaboost

Stacking

使用多个不同的分类器对训练集进预测，把预测得到的结果作为一个次级分类器的输入。次级分类器的输出是整个模型的预测结果。



Stacking



Stacking

总结

“人多力量大”
好的集成学习：多样性+准确性

	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

	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

	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