目录

个体与集成

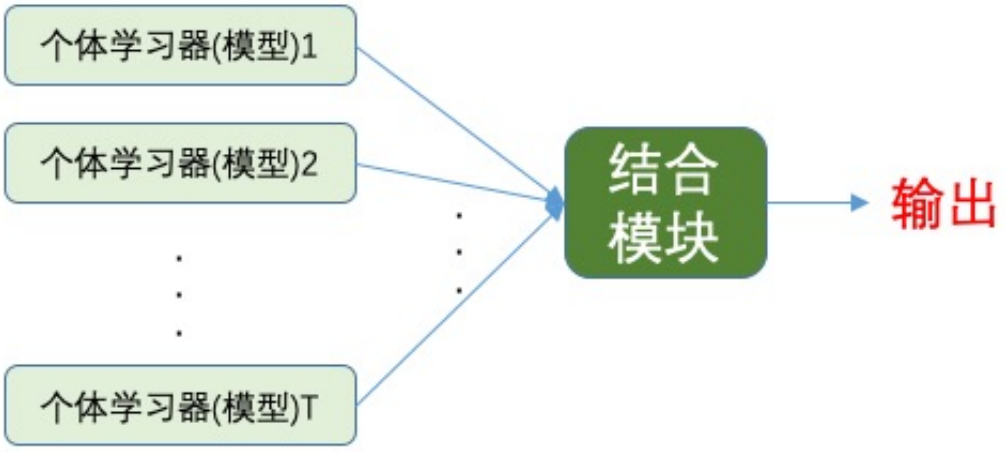
【参考】

- quantdare - What is the difference between Bagging and Boosting?

概念：多分类器（模型）系统（multi-classifier system）、基于委员会的学习（committee-based learning）、个体学习器（individual learner）、同质（homogeneous）、基学习器（base learner）、基学习算法（base learning algorithm）、异质（heterogeneous）、组件学习器（Component learner）、泛化性能、弱学习器（weak learner）、投票法（voting）、和而不同、多样性（diversity）、串行（序列化）、并行、

集成学习（ensemble learning）通过构建并结合多个**学习器（模型）**来完成学习的任务，有时候也被成为**多分类器（模型）系统（multi-classifier system）、基于委员会的学习（committee-based learning）**。

如下图，集成学习的一般结构：



首先产生一组**个体学习器（individual learner）**，再用某种策略将他们集合起来。

个体学习器通常由一个现有的**学习算法**从训练数据中训练而来，例如 C4.5决策树算法，BP 神经网络算法，此时集成中只包含同种类型的个体学习器，这样的集成是**同质（homogeneous）**的。同质集成中的个体学习器又称为**基学习器（base learner）**，相应的算法也成为**基学习算法（base learning algorithm）**。

集成也可以包含不同的个体学习器，例如同时包含决策树和神经网络，这样的集成是**异质（heterogeneous）**的。异质集成中的个体学习器由不同的学习算法生成，这时不再有基学习算法，相应的，个体学习器一般不称为基学习器，常称为**组件学习器（Component learner）**或者直接称为个体学习器。

集成学习通过将多个学习器进行结合，常可获得比单一学习器显著优越的泛化性能。这对**弱学习器（weak learner）**尤为明显，弱学习器是指泛化性能略优于随机猜测的学习器，如二分类问题上精度略高于50%的分类器，因此集成学习的很多理论研究都是针对弱学习器进行的，而基学习器有时也被称作弱学习器。但在实践中，常用的还是强学习器。

把好东西与坏东西掺在一起，通常结果是比最差的要好，但比最好的要差。那么把多个学习器结合起来，如何获得比单一学习器更好的性能呢。以二分类为例：有三个分类器，在三个测试样本上的表现如下图：

	样本1	样本2	样本3
h1	√	√	×
h2	×	√	√
h3	√	×	√
集成	√	√	√

(a) 集成性能提升

	样本1	样本2	样本3
h1	√	√	×
h2	√	√	×
h3	√	√	×
集成	√	√	×

(b) 集成不起作用

	样本1	样本2	样本3
h1	√	×	×
h2	×	√	×
h3	×	×	√
集成	×	×	×

(c) 集成起负作用

其中 √ 表示真确分类，×表示错误分类，h1~h3表示分类器，集成的结果通过**投票法（voting）**产生，即**少数服从多数**。 可以看到 a 的每一个分类器精度为66.6%，但集成之后的结果为 100%，集成让性能得到了提升；同理 b 集成不起作用，而 c 集成起到了反作用。

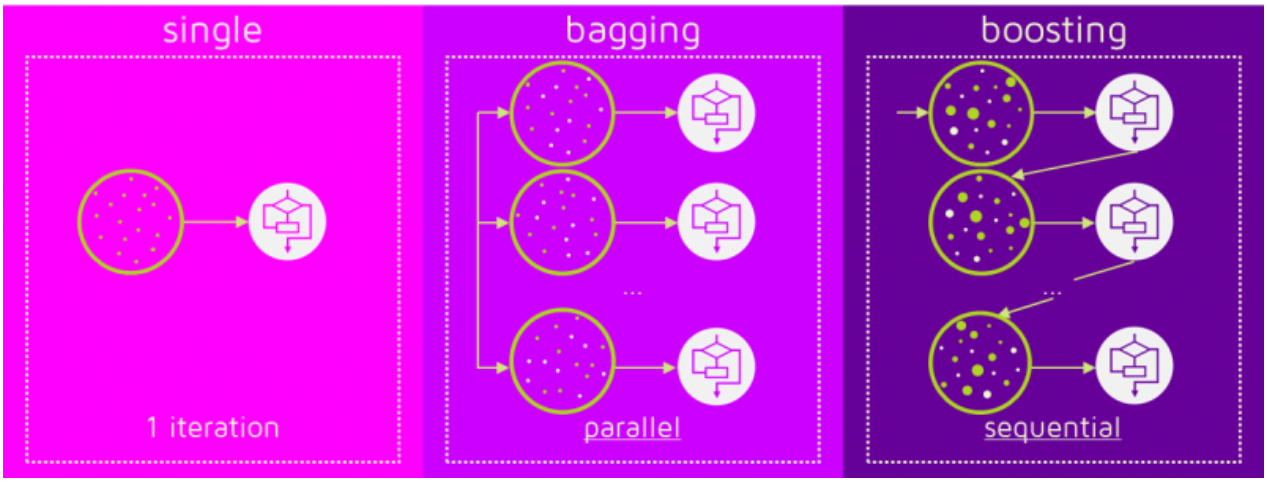
因此要想获得好的结果：**个体集学习器应该“和而不同”，即个体学习器要有一个定的“准确性”，即学习器不能太坏，并且要有“多样性（diversity）”，即学习器之间要有差异。**

然而个体学习器的准确性和多样性本身就存在着冲突，一般，准确性提高之后，要增加多样性就需要牺牲准确性。事实上，如何产生并结合好而不同的个体学习器，恰是集成学习的研究核心。

根据个体学习器的生成方式的不同，目前集成学习可以分为两大类：

- 个体学习器之间存在强依赖关系，必须串行生成的序列化方法，此代表是 Boosting
- 个体学习器之间不存在强依赖关系，可以同时生成的并行化方法，此代表为 Bagging和 随机森林（Random Forest）

Bagging Boosting 示意图如下：



Boosting

【附加】

- Greedy Function Approximation: A Gradient Boosting Machine **概念：样本分布**

Boosting 是**一族**可将若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一个基学习器，在根据基学习器的表现对**训练样本的分布**进行调整，使得先前基学习器做错的训练样本在后续得到更多关注，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，直到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**。

Boosting 族算法最著名的代表是 AdaBoost（Adaptive Boosting 自适应增强）。

AdaBoost 算法

概念：符号函数、指示函数、指数损失函数（exponential loss function）、

【参考】

- csdn - Adaboost算法原理分析和实例+代码（简明易懂）
- sklearn - AdaBoost
- paper - A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting, 1997 加性模型(additive model)

为了讨论方便，做出以下符号约定：

- D 表示训练集；

- $\boldsymbol{x}$ 表述训练集的样本
- $y_i \in \{-1, +1\}$ 表示样本的可能标签；
- T 表示需要训练的次数，即需要训练多少个基学习器；
- f 是真是函数；
- h_t 表示第 t 轮的基学习器即弱学习器；
- α_t表示第 t 个基学习器的权重；
- H_i表示前 i 轮基学习器的加权组合
- H 表示最终的强分类器；
- $\mathcal{D}$表示训练集中样本权重分布，初始时的所有样本的权重值都为$\frac{1}{m}$，m 为样本的个数；
- $\mathcal{D}_t$ 表示在第 t 轮训练时，训练集中样本的权重分布，如$\mathcal{D}_1 = \frac{1}{m}$；
- ω_{ti} 表示在 t 论，第 i 个样本的权重
- Z_t 是规范化因子，用于确保$\mathcal{D}_t$是一个分布
- $sign(y)$ 表示**符号函数**，当值 y<0时取值为 -1，当 y=0时取值为 0，当 y>0时取值为 1.
- $\mathbf{I}(y)$ 表示**指示函数**，当 y 为真的时候值为1，假时值为 0

一下算法流程是基于加性模型的算法，即依据的是 Freund and Schapire 1997 的论文，在 sklearn 中是基于 SAMME 和 SAMME.R 算法实现，在更新学习器权重与样本权重时有些许差别，且后者可以用于多分类系统。

算法流程

输入：
训练集：$D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$，含有 m 个样本
基学习算法：$\mathcal{L}$
训练轮数：T

过程：
1.这是初始化的数据集样本权重分布，$\mathcal{D}_1 = \frac{1}{m}$
2.进行迭代 **for t =1,2,...T do**:
(a) 训练基学习器：$h_t = \mathcal{L}(D, \mathcal{D}_t)$
(b) 计算基学习器的错误率：$\varepsilon_t = P_{\boldsymbol{x} \sim \mathcal{D}_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$，即 $\varepsilon_t = P_{\boldsymbol{x} \sim \mathcal{D}_t}(h_t(\boldsymbol{x}) \neq y_i)$，也就是：

$$\varepsilon_t = P_{\boldsymbol{x} \sim \mathcal{D}_t}(h_t(\boldsymbol{x}) \neq y_i) = \sum_{i=1}^{m} \omega_{ti} \mathbf{I}(H_t(x_i) \neq y_i) \tag{A1}$$

(c) 检查条件： **if** $\varepsilon_t > 0.5$ **then break**，检测训练出来的基学习器是否好于随机分类器，不好于则退出学习
(d) 设置基学习器权重：若训练出来的基学习器没有好于随机分类器，为当前基学习器设置权重

$$\alpha_t = \frac{1}{2}\ln(\frac{1-\varepsilon_t}{\varepsilon_t}) \tag{A2}$$

(e) 更新样本权重：$Z_t = 2\sqrt{\varepsilon_t(1-\varepsilon_t)}$

$$\begin{aligned} \mathcal{D}_{t+1}(\boldsymbol{x}) &= \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t} \times \begin{cases} exp(-\alpha_t) & if\ h_t(\boldsymbol{x}) = f(\boldsymbol{x}) \\ exp(\alpha_t) & if\ h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}) \end{cases} \\ &= \frac{\mathcal{D}_t(\boldsymbol{x}) exp\left(-\alpha_t f(\boldsymbol{x}) h_t(\boldsymbol{x})\right)}{Z_t} \end{aligned} \tag{A3}$$

输出：

$$H(\boldsymbol{x}) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right) \tag{A4}$$

样本权值的更新

从式子（A3）可以看到样本权重的更新时依赖于α_t和Z_t，而α_t从式子（A2）可以看出是依赖于ε_t，Z_t也是，因此权重的更新时依赖于ε_t。而权重更新的公式又可以写成如下的形式：

$$\mathcal{D}_{t+1}(\boldsymbol{x}) = \frac{\mathcal{D}_t(\boldsymbol{x}) exp\left(-\alpha_t y_i h_t(x_i)\right)}{Z_t} \tag{A5}$$

基学习器在与测试有以下两种情况：
(1) 对样本预测错误：即 y=1是预测为了-1，当y=-1时预测成了 1，因此$y_i h_t(x_i) = -1$，这时有：

$$\begin{aligned} \mathcal{D}_{t+1}(\boldsymbol{x}) &= \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t}\exp(\alpha_t) = \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t}\exp(\frac{1}{2}\ln(\frac{1-\varepsilon_t}{\varepsilon_t})) \\ &= \frac{\mathcal{D}_t(\boldsymbol{x})}{2\sqrt{\varepsilon_t(1-\varepsilon_t)}}\sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}} = \frac{\mathcal{D}_t(\boldsymbol{x})}{2\varepsilon_t} \end{aligned} \tag{A6}$$

(2) 当样本预测正确的时候，即 y=1是预测为了1，当y=-1时预测成了 -1，因此$y_i h_t(x_i) = 1$，这时有：

$$\begin{aligned} \mathcal{D}_{t+1}(\boldsymbol{x}) &= \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t}\exp(-\alpha_t) = \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t}\exp(-\frac{1}{2}\ln(\frac{1-\varepsilon_t}{\varepsilon_t})) \\ &= \frac{\mathcal{D}_t(\boldsymbol{x})}{2\sqrt{\varepsilon_t(1-\varepsilon_t)}}\sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}} = \frac{\mathcal{D}_t(\boldsymbol{x})}{2(1-\varepsilon_t)} \end{aligned} \tag{A7}$$

因此我们就可以得到如下的权值更新公式：

- 当样本分类错误时，样本权值更新为：$\mathcal{D}_{t+1}(\boldsymbol{x}) = \frac{\mathcal{D}_t(\boldsymbol{x})}{2\varepsilon_t}$
- 当样本分类正确时，样本权值更新为：$\mathcal{D}_{t+1}(\boldsymbol{x}) = \frac{\mathcal{D}_t(\boldsymbol{x})}{2(1-\varepsilon_t)}$

学习器权值的更新

AdaBoost 有多种推到方法，最简单的是基于**加性模型（additive model）**，即基学习器的线性组合：

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

用上式最小化**指数损失函数（exponential loss function）**：

$$\ell_{exp}(H \mid \mathcal{D}) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H(\boldsymbol{x})}\right] \tag{2}$$

在 AdaBoost 算法中，第一个基分类器h_1是通过直接将基本学习算法用于初始数据的分布得到。之后迭代生成 h_t 和α_t，当基分类器 h_t 基于分布 $\mathbf{D}_t$产生之后，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：

$$\begin{aligned}\ell_{exp}(\alpha_t h_t \mid \mathcal{D}) &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[e^{-f(\boldsymbol{x})\alpha_t h_t(\boldsymbol{x})}\right] \\ &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[e^{-\alpha_t}\mathbf{I}(f(\boldsymbol{x})=h_t(\boldsymbol{x}))+e^{\alpha_t}\mathbf{I}(f(\boldsymbol{x})\neq h_t(\boldsymbol{x}))\right] \\ &= e^{-\alpha_t}P_{\boldsymbol{x}\sim\mathcal{D}_t}(f(\boldsymbol{x})=h_t(\boldsymbol{x}))+e^{\alpha_t}P_{\boldsymbol{x}\sim\mathcal{D}_t}(f(\boldsymbol{x})\neq h_t(\boldsymbol{x})) \\ &= e^{-\alpha_t}(1-\varepsilon_t)+e^{\alpha_t}\varepsilon_t\end{aligned} \tag{3}$$

其中$\varepsilon_t = P_{\boldsymbol{x}\sim\mathcal{D}_t}(f(\boldsymbol{x})\neq h_t(\boldsymbol{x}))$，令指数损失行数的导数为：

$$\frac{\partial \ell_{exp(\alpha_t h_t \mid \mathcal{D})}}{\partial \alpha_t} = e^{-\alpha_t}(1-\varepsilon_t)+e^{\alpha_t}\varepsilon_t \tag{4}$$

令（4）式为零就可以得到：

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$$

多分类

【参考】

- paper - J. Zhu, H. Zou, S. Rosset, T. Hastie, “Multi-class AdaBoost”, 2009. SAMME 与 SAMME.R 算法
- stackoverflow - Why estimator_weight in SAMME.R AdaBoost algorithm is set to 1

AdaBoost 算法只能用于二分类，经过改进的 SAMME 与 SAMME.R 算法则可以用于多分类系统。

符号约定：

- n：表示样本个数
- K：表示类别总数，当 K 为 2 时就是二分类
- C：表示输出类别函数,即分类器的加权和
- c_i：表示第 i 个样本的类别
- M：表示学习器的个数
- α：表示学习器权重
- ω：表示样本权重
- T：分类器

SAMME 算法

算法流程

1.初始化观测权重 $\omega_i = \frac{1}{n}, i = 1, 2, \ldots, n$

2.**for m=1,2,...,M DO:**

(a) 使用权重 ω_i 在训练数据上训练分类器 $T^{(m)}(\boldsymbol{x})$

(b) 计算分类器误差：

$$\epsilon^{(m)} = \sum_{i=1}^{n} \omega_i \mathbf{I}\left(c_i \neq T^{(m)}(x_i)\right) / \sum_{i=1}^{n} \omega_i$$

(c) 计算当前分类器权重：

$$\alpha^{(m)} = \log \frac{1-\epsilon^{(m)}}{\epsilon^{(m)}} + \log(K-1)$$

(d) 更新样本权重：

$$\omega_i \leftarrow \omega_i \cdot \exp\left(\alpha^{(m)} \cdot \mathbf{I}\left(c_i \neq T^{(m)}(x_i)\right)\right), i = 1, \ldots, n$$

(e) 重新归一化样本权重 ω_i

3.**输出**

$$C(\boldsymbol{x}) = arg\max_{k} \sum_{m=1}^{M} \alpha^{(m)} \cdot \mathbf{I}\left(T^{(m)}(\boldsymbol{x}) = k\right)$$

SAMME.R 算法

此处的 R 是 Real 的意思，SAMME.R 使用了概率评估（probability estimates）方法来更新加性模型（additive model），而 SAMME 只使用了分类。

1.初始化观测权重 $\omega_i = \frac{1}{n}, i = 1, 2, \ldots, n$

2.**for m=1,2,...,M DO:**

(a) 使用权重 ω_i 在训练数据上训练分类器 $T^{(m)}(\boldsymbol{x})$

(b) 获取权重化类别的概率评估：

$$p_k^{(m)}(\boldsymbol{x}) = Prob_{\omega}(c = k|\boldsymbol{x}), k = 1, \ldots, K$$

(c) 更新：

$$h_k^{(m)} \leftarrow (K-1)\left(\log p_k^{(m)}(\boldsymbol{x}) - \frac{1}{K}\sum_{k'} \log p_{k'}^{(m)}(\boldsymbol{x})\right), k = 1, \ldots, K$$

(d) 更新样本权重：

$$\omega_i \leftarrow \omega_i \cdot \exp\left(-\frac{K-1}{K} y_i^T \log p^m(x_i)\right), i = 1, \ldots, n$$

(e) 重新归一化样本权重 ω_i

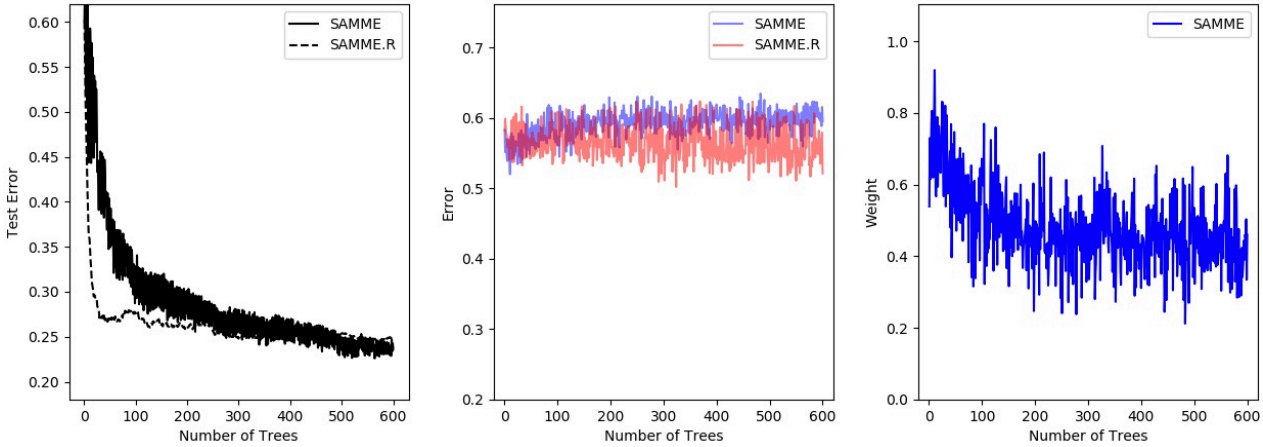
3.**输出**

$$C(\boldsymbol{x}) = arg\max_k \sum_{m=1}^{M} h_k^{(m)}(\boldsymbol{x})$$

SAMME 与 SAMME.R 的比较

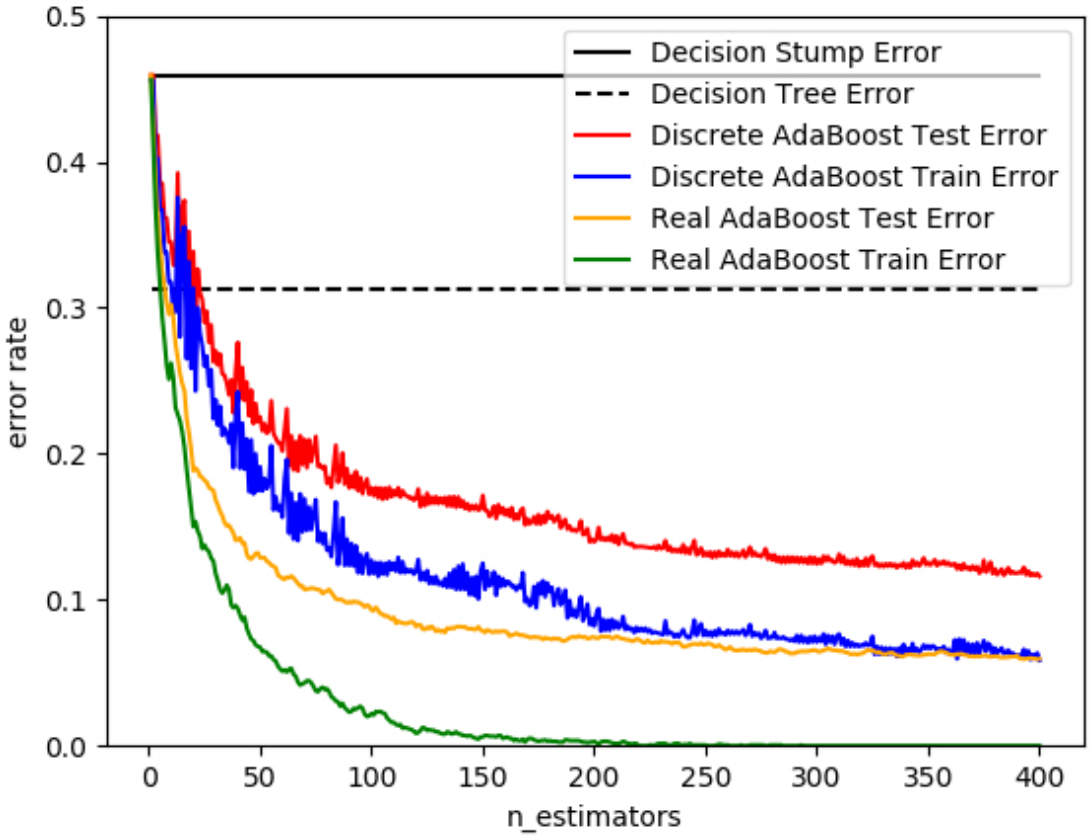
【参考】

- sklearn - Multi-class AdaBoosted Decision Trees



由上面左侧的图可以看到，使用了概率评估的 SAMME.R 收敛的速度比只是用分类的 SAMME 方法要快很多，而且也获得了更低的测试误差（test error），且适应的迭代次数更少。中间图是每个树在测试集额上的分类误差（classification error）。右侧图展示了 SAMME 中每棵树的权重变化，SAMME.R 不需要权重，因此没有展示。

下图展示的是两者在测试与训练是的误差比较，其中 Discrete 是 SAMME，Real 是 SAMME.R。



AdaBoost 案例

【参考】

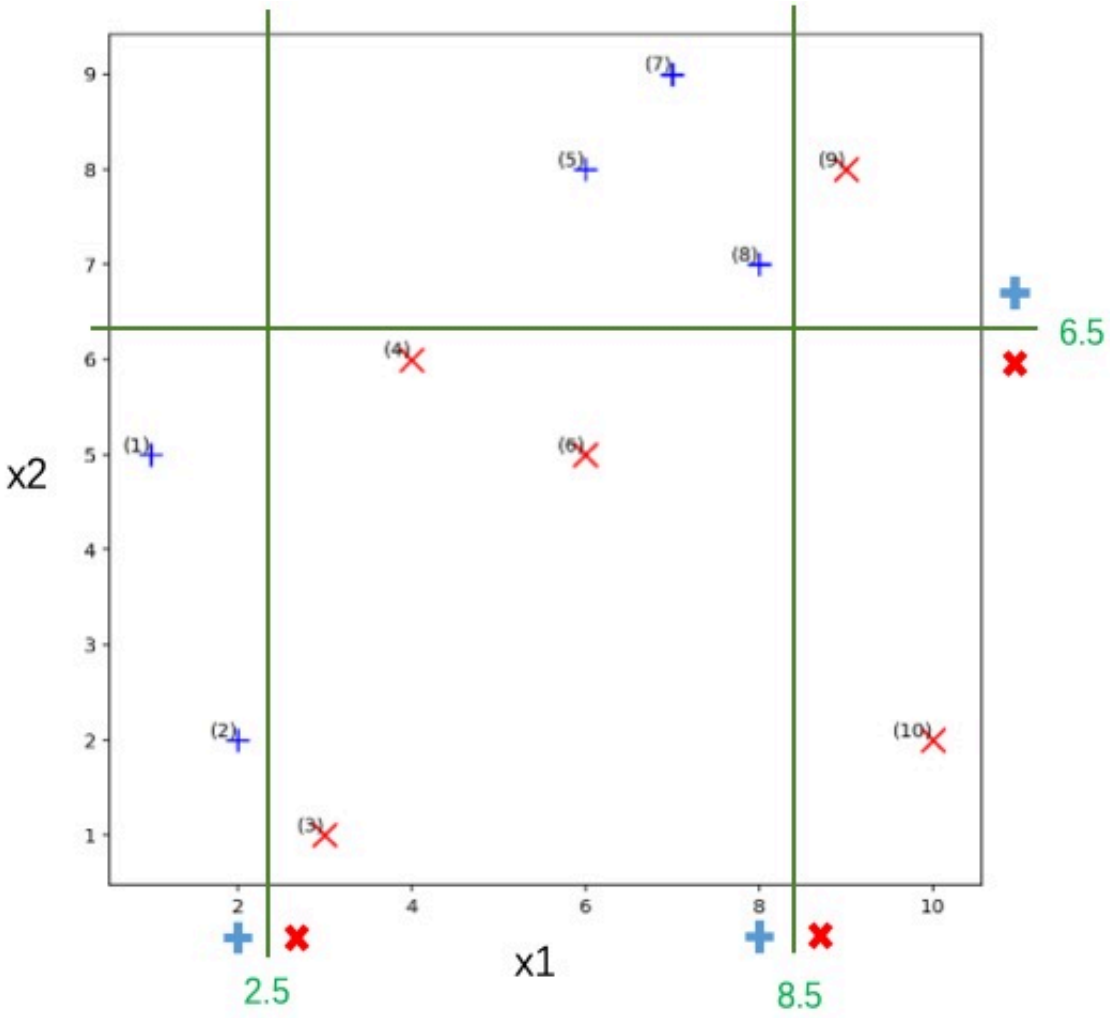
- csdn - Adaboost 算法的原理与推导 较为全面

下面的案例，查看上参考文章更容易理解，因为下面的例子直接就给出了h_1, h_2, h_3三个分类器，但这三个分类器其实是根据上面提到的AdaBoost算法在不同样本权重下训练出来的。可以参考 sklearn 的 AdaBoost 源码实现。

有如下数据：

样本序号	1	2	3	4	5	6	7	8	9	10
样本点X	(1,5)	(2,2)	(3,1)	(4,6)	(6,8)	(6,5)	(7,9)	(8,7)	(9,8)	(10,2)
类别 Y	1	1	-1	-1	1	-1	1	1	-1	-1

将这个 10 条数据分为两类，图中 + 表示类别为 1 ， x 表示类别为 -1.使用水平和垂直执行作为分类器，下图有三个弱分类器：



$$h_1 = \begin{cases} 1 & X_1 < 2.5 \\ -1 & X_1 > 2.5 \end{cases} \quad h_2 = \begin{cases} 1 & X_1 < 8.5 \\ -1 & X_1 > 8.5 \end{cases} \quad h_3 = \begin{cases} 1 & X_2 < 6.5 \\ -1 & X_2 > 6.5 \end{cases}$$

初始化

首先需要初始化训练样本数据的权值分布，每一个训练样本最开始时都被赋予相同的权值：$\omega_{1i} = \frac{1}{m}$，这样训练样本集的初始权值分布$D_1(i)$： 令每个权值$\omega_{1i} = \frac{1}{m} = 0.1$，其中，m = 10，i = 1,2, ..., 10，然后分别对于t= 1,2,3, ...等值进行迭代（t表示迭代次数，表示第t轮），下表已经给出训练样本的权值分布情况：

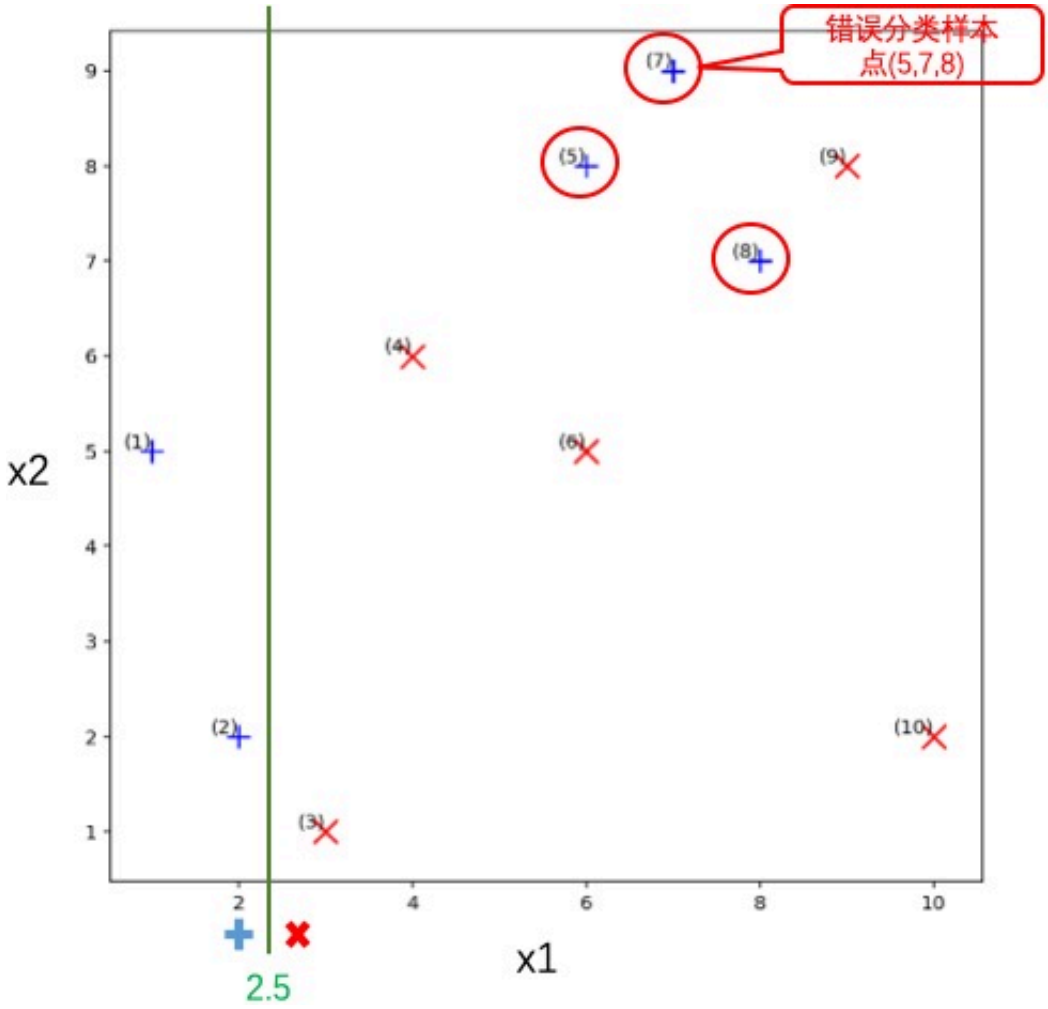
样本序号	1	2	3	4	5	6	7	8	9	10
样本点X	(1,5)	(2,2)	(3,1)	(4,6)	(6,8)	(6,5)	(7,9)	(8,7)	(9,8)	(10,2)
类别 Y	1	1	-1	-1	1	-1	1	1	-1	-1

第一次迭代

初试的权值分布$D_1 = 1/m$（10个数据，每个数据的权值皆初始化为0.1），
$D_1 = [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]$

在权值分布D1的情况下，取已知的三个弱分类器r_1、r_2、r_3中误差率最小的分类器作为第1个基本分类器$h_1(\boldsymbol{x})$（三个弱分类器的误差率都是0.3，此处选择第一个）。错误率等于该分类器被错分样本的权重之和，因为有 3 个被分错，且每个样本的权重都为 0.1，因此可以计算出误差误差率和h_1需要更新的权重：

$$\varepsilon_1 = (0.1 + 0.1 + 0.1) = 0.3$$
$$\alpha_1 = \frac{1}{2}\ln(\frac{1-\varepsilon_1}{\varepsilon_1}) = \frac{1}{2}\ln(\frac{1-0.3}{0.3}) = 0.4236$$



之后，更新训练样本的权值分布，用于下一轮的迭代，对于正确分类的训练样本{1,2,3,4,6,9,10}（共7个）的权值更新使用公式（A7）：

$$\mathcal{D}_2 = \frac{\mathcal{D}_1}{2(1-\varepsilon_1)} = \frac{1}{10} \times \frac{1}{2(1-0.3)} = \frac{1}{14}$$

可以看到，**正确分类的样本的权值由原来的1/10减少到了1/14**。

对于错误分类的样本{5,7,8}的权值更新使用公式（A6）进行更新：

$$\mathcal{D}_2 = \frac{\mathcal{D}_1}{2\varepsilon_1} = \frac{1}{10} \times \frac{1}{2 \times 0.3} = \frac{1}{6}$$

可以看到，**错误分类的样本的权值由原来的1/10增加到了1/6**.

经过第一轮的迭代之后，各个样本的权值变为了：
$D_2 = [1/14, 1/14, 1/14, 1/14, 1/6, 1/14, 1/6, 1/6, 1/14, 1/14]$.
由于样本数据{5,7,8}被$h_1(x)$分错了，所以它们的权值由之前的0.1增大到1/6；反之，其它数据皆被分正确，所以它们的权值皆由之前的0.1减小到1/14，下表给出了权值分布的变换情况：

样本序号	1	2	3	4	5	6	7	8	9	10
样本点X	(1,5)	(2,2)	(3,1)	(4,6)	(6,8)	(6,5)	(7,9)	(8,7)	(9,8)	(10,2)
类别 Y	1	1	-1	-1	1	-1	1	1	-1	-1
权值分布 D1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
权值分布 D2	1/14	1/14	1/14	1/14	1/6	1/14	1/6	1/6	1/14	1/14
sign(H1(x))	1	1	-1	-1	-1	-1	-1	-1	-1	-1

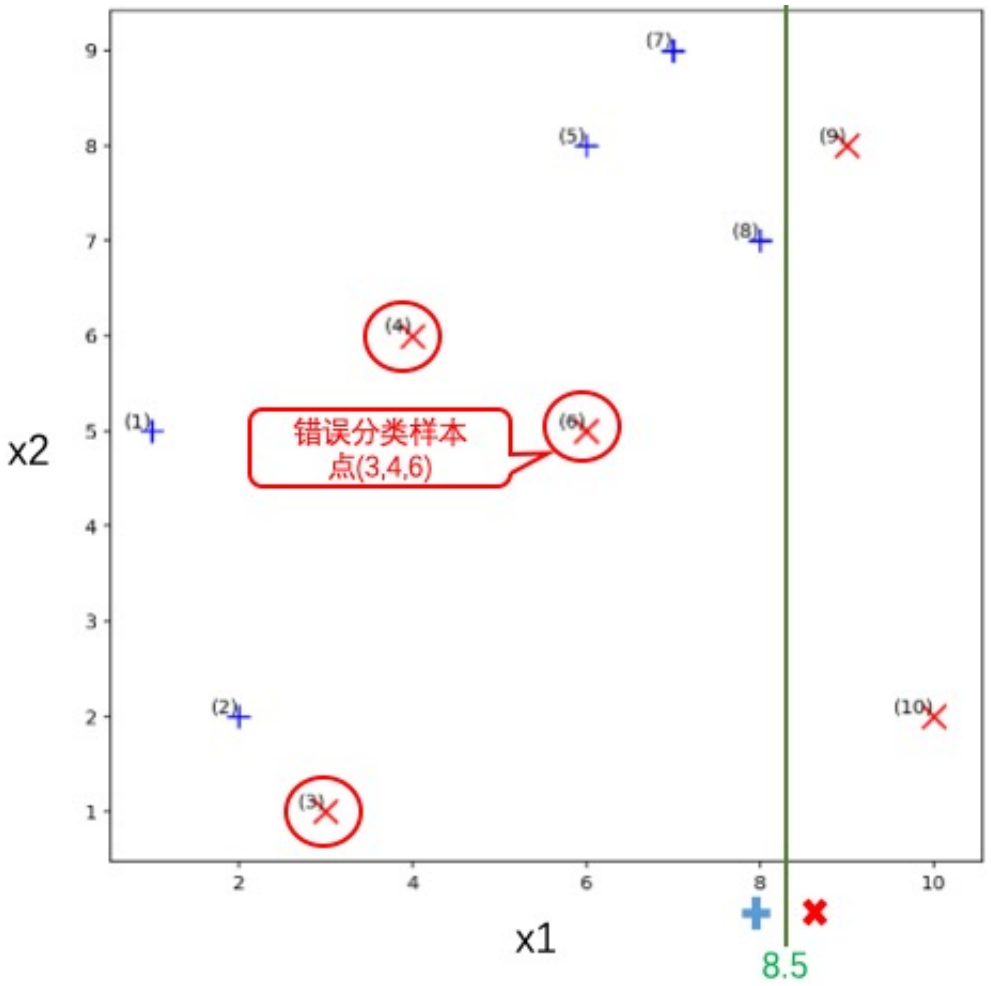
被选中的红色列，是被$h_1(\boldsymbol{x})$ 分错的样本，没有选择中的知识被真确分类的样本。因此分类函数为：

$$H_1(\boldsymbol{x}) = \alpha_1 h_1(\boldsymbol{x}) = 0.4236 h_1(\boldsymbol{x})$$

此时，组合一个基本分类器$sign(H_1(x))$作为强分类器在训练数据集上有3个误分类点（即5,7,8），此时强分类器的训练错误为：0.3

第二次迭代

在权值分布$\mathcal{D}_2$的情况下，再取三个弱分类器r_1、r_2、r_3中误差率最小的分类器作为第2个基本分类器$h_2(x)$：
① 当取弱分类器h1=X1=2.5时，此时被错分的样本点为{5,7,8}”：
误差率e=1/6+1/6+1/6=3/6=1/2；
② 当取弱分类器h2=X1=8.5时，此时被错分的样本点为{3,4,6}：
误差率e=1/14+1/14+1/14=3/14；
③ 当取弱分类器h3=X2=6.5时，此时被错分的样本点为{1,2,9}：
误差率e=1/14+1/14+1/14=3/14；



因此，取当前最小的分类器r_2作为第2个基本分类器$h_2(x)$。显然，$h_2(x)$把样本{3,4,6}分错了，根据$\mathcal{D}_2$可知它们的权值为$\omega_{23} = 1/14, \omega_{24} = 1/14, \omega_{26} = 1/14$，所以 $h_2(x)$ 在训练数据集上的误差率：

$$\varepsilon_2 = P(h_2(x_i) \neq y_i) = (1/14 + 1/14 + 1/14) = 3/14$$

$$\alpha_2 = \frac{1}{2}\ln(\frac{1-\varepsilon_2}{\varepsilon_2}) = 0.6496$$

之后，更新训练样本的权值分布，用于下一轮的迭代，对于正确分类的训练样本的权值更新使用公式（A7）：

$$\mathcal{D}_3 = \frac{\mathcal{D}_2}{2(1-\varepsilon_2)} = \frac{7}{11}\mathcal{D}_2$$

对于错误分类的样本的权值更新使用公式（A6）进行更新：

$$\mathcal{D}_3 = \frac{\mathcal{D}_2}{2\varepsilon_2} = \frac{7}{3}\mathcal{D}_2$$

新的样本的权重分布为：$D_3 = [1/22, 1/22, 1/6, 1/6, 7/66, 1/6, 7/66, 7/66, 1/22, 1/22]$。下表给出了权值分布的变换情况：

样本序号	1	2	3	4	5	6	7	8	9	10
样本点X	(1,5)	(2,2)	(3,1)	(4,6)	(6,8)	(6,5)	(7,9)	(8,7)	(9,8)	(10,2)
类别 Y	1	1	-1	-1	1	-1	1	1	-1	-1
权值分布 D1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
权值分布 D2	1/14	1/14	1/14	1/14	1/6	1/14	1/6	1/6	1/14	1/14
sign(H1(x))	1	1	-1	-1	-1	-1	-1	-1	-1	-1
权值分布 D3	1/22	1/22	1/6	1/6	7/66	1/6	7/66	7/66	1/22	1/22
sign(H2(x))	1	1	1	1	1	1	1	1	-1	-1

浅紫色是被 $h_2(\boldsymbol{x})$分错的样本。

可得分类函数：$H_2(x) = 0.4236h_1(\boldsymbol{x}) + 0.6496h_2(\boldsymbol{x})$。此时，组合两个基本分类器$sign(H_2(\boldsymbol{x}))$作为强分类器在训练数据集上有3个误分类点（即{3,4,6}），此时强分类器的训练错误为：0.3

第三次迭代

在权值分布$\mathcal{D}_3$的情况下，再取三个弱分类器r_1、r_2、r_3中误差率最小的分类器作为第3个基本分类器$h_3(x)$：

① 当取弱分类器h1=X1=2.5时，此时被错分的样本点为{5,7,8}：

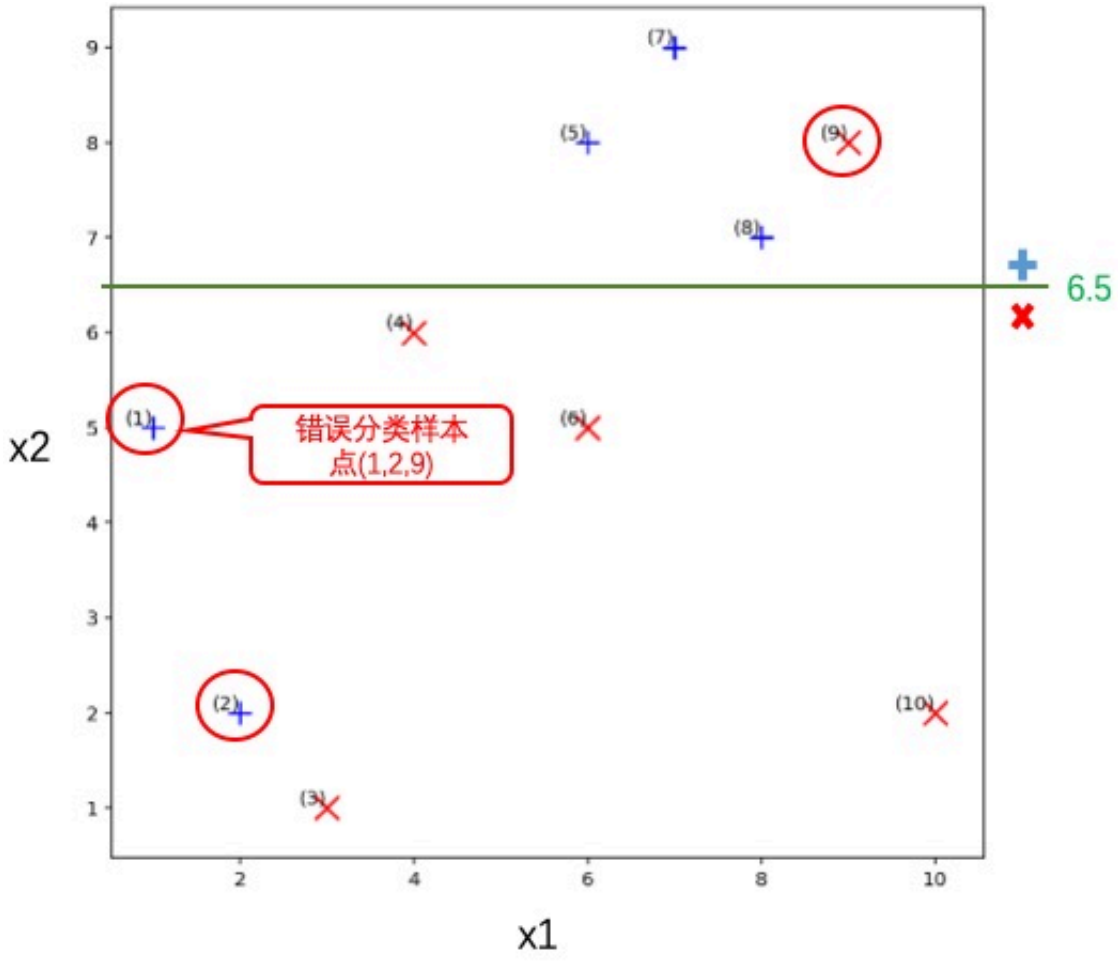
误差率e=7/66+7/66+7/66=7/22；

② 当取弱分类器h2=X1=8.5时，此时被错分的样本点为{3,4,6}：

误差率e=1/6+1/6+1/6=1/2=0.5；

③ 当取弱分类器h3=X2=6.5时，此时被错分的样本点为{1,2,9}：

误差率e=1/22+1/22+1/22=3/22；



因此，取当前最小的分类器r_3作为第3个基本分类器$h_3(x)$，此时被错误分类的样本为{1,2,9}，可以知道他们的权值为：$\omega_{31}=1/14,\omega_{32}=1/14,\omega_{39}=1/14$，所以 $h_3(x)$ 在训练数据集上的误差率：

$$\varepsilon_3 = P(h_2(x_i)\neq y_i) = (1/22+1/22+1/2) = 3/22$$
$$\alpha_3 = \frac{1}{2}\ln(\frac{1-\varepsilon_3}{\varepsilon_3}) = 0.9229$$

之后，更新训练样本的权值分布，用于下一轮的迭代，对于正确分类的训练样本的权值更新使用公式（A7）：

$$\mathcal{D}_4 = \frac{\mathcal{D}_3}{2(1-\varepsilon_3)} = \frac{11}{19}\mathcal{D}_3$$

对于错误分类的样本的权值更新使用公式（A6）进行更新：

$$\mathcal{D}_4 = \frac{\mathcal{D}_3}{2\varepsilon_3} = \frac{11}{3}\mathcal{D}_3$$

这样，第3轮迭代后，得到各个样本数据新的权值分布为：
$\mathcal{D}_4 = [1/6, 1/6, 11/114, 11/114, 7/114, 11/114, 7/114, 7/114, 1/6, 1/38]$，下表给出了权值分布的变换情况：

样本序号	1	2	3	4	5	6	7	8	9	10
样本点X	(1,5)	(2,2)	(3,1)	(4,6)	(6,8)	(6,5)	(7,9)	(8,7)	(9,8)	(10,2)
类别 Y	1	1	-1	-1	1	-1	1	1	-1	-1
权值分布 D1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
权值分布 D2	1/14	1/14	1/14	1/14	1/6	1/14	1/6	1/6	1/14	1/14
sign(H1(x))	1	1	-1	-1	-1	-1	-1	-1	-1	-1
权值分布 D3	1/22	1/22	1/6	1/6	7/66	1/6	7/66	7/66	1/22	1/22
sign(H2(x))	1	1	1	1	1	1	1	1	-1	-1
权值分布 D4	1/6	1/6	11/114	11/114	7/114	11/114	7/114	7/114	1/6	1/38
sign(H3(x))	1	1	-1	-1	1	-1	1	1	-1	-1

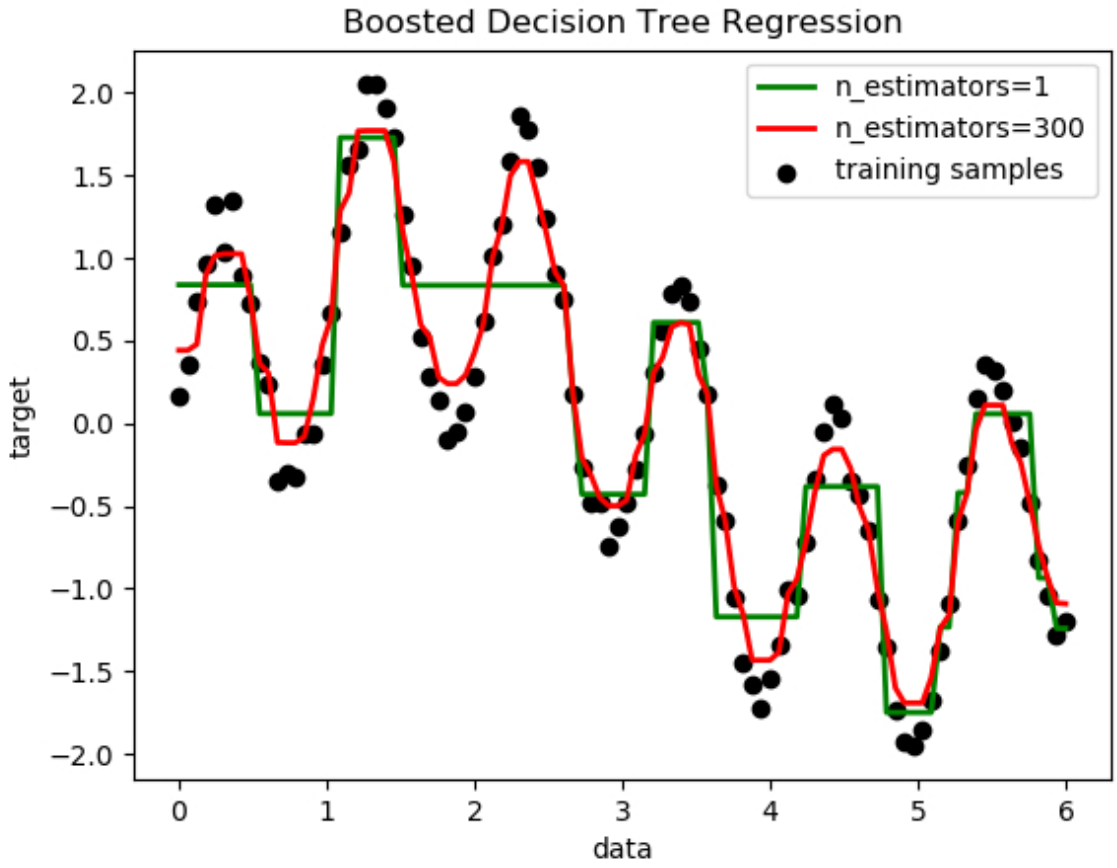
可得分类函数：$H_3(x) = 0.4236h_1(\boldsymbol{x}) + 0.6496h_2(\boldsymbol{x}) + 0.9229h_3(\boldsymbol{x})$。此时，组合三个基本分类器sign(f3(x))作为强分类器，在训练数据集上有0个误分类点。至此，整个训练过程结束。

整合所有分类器，可得最终的强分类器为：

$$H(\boldsymbol{x}) = \sum_{t=1}^{T}\alpha_t h_t(\boldsymbol{x}) = 0.4236h_1(\boldsymbol{x}) + 0.6496h_2(\boldsymbol{x}) + 0.9229h_3(\boldsymbol{x})$$

sklearn 案例

- sklearn - Decision Tree Regression with AdaBoost



AdaBoost 特点

AdaBoost主要有以下两个特点：

- （1）是训练的错误率上界，随着迭代次数的增加，会逐渐下降；
- （2）是Adaboost算法即使训练次数很多，也不会出现过拟合的问题。

标准的 AdaBoost 只可用于二分类任务，遇到多分类的情况，需要借助 one-versus-rest 的思想来训练多分类模型。

优点

- （1）Adaboost提供一种框架，在框架内可以使用各种方法构建子分类器。可以使用简单的弱分类器，不用对特征进行筛选，也**不存在过拟合的现象**。
- （2）Adaboost算法不需要弱分类器的先验知识，最后得到的强分类器的分类精度依赖于所有弱分类器。无论是应用于人造数据还是真实数据，Adaboost都能显著的提高学习精度。
- （3）Adaboost算法不需要预先知道弱分类器的错误率上限，且最后得到的强分类器的分类精度依赖于所有弱分类器的分类精度，可以深挖分类器的能力。Adaboost可以根据弱分类器的反馈，自适应地调整假定的错误率，执行的效率高。
- （4）Adaboost对同一个训练样本集训练不同的弱分类器，按照一定的方法把这些弱分类器集合起来，构造一个分类能力很强的强分类器，即“三个臭皮匠赛过一个诸葛亮”。

缺点：
在Adaboost训练过程中，Adaboost会使得难于分类样本的权值呈指数增长，训练将会过于偏向这类困难的样本，导致Adaboost算法易受噪声干扰。此外，Adaboost依赖于弱分类器，而弱分类器的训练时间往往很长。

Booting 总结

Boosting 算法要求基学习器能对特定的数据分布学习，这可通过**重赋权法（re-weighting）**实施，即在训练过程中的每一轮，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采样，再用重新采样而得的样本集对基学习器进行训练。
一般而言，这两种做法没有明显的优劣之分。

Boosting 在每一轮都要检测当前生成的基学习器是否满足基本条件，如算法中（c）行，**检查当前基分类器是否比随机猜测的好**，一旦条件不满足，则当前基学习器即被抛弃，同时学习过程停止。这种情况下，初始设置的学习轮数 T 也许还没有达到，可能最终导致最终集成中只包含很少的基学习器而性能不佳。若采用重采样法，则可以获得的重启动的机会以避免训练过程中的过早停止，即在不满抛弃不满足条件的当前基学习器之后，可根据当前分布重新对训练样本进行采样，在基于新的采样结果从新训练学习器，从而使得学习的过程可以持续到特定的轮数。

从偏差-方差分解来看，Boosting 主要关注**降低偏差**，因此Boosting 能基于泛化能力相当弱的学习器构建出很强的集成。