集成模型之Adaboost算法(三)

上一节课我们介绍了Boosting算法的原理，AdaBoost算法和提升树(boosting tree)系列算法是Boosting算法最著名的应用之一。这节课我们主要介绍以下AdaBoost算法。

1.1 Adaboost 三个问题

AdaBoost 的全称为Adaptive Boosting。之所以称为 Adaptive 是由于下一轮分类器的训练样本权重会随上一轮分类器的分类效果做相应的调整。基于之前学过的 Boosting 算法，整个 Adaboost 迭代算法可以简化为以下三步：

1. 初始化训练样本的权值 D_0 。则每一个训练样本的权重被初始化为 $\frac{1}{m}$ ，其中 m 为样本的数量。
2. 迭代训练弱分类器。在迭代过程中，需要对样本权重 D_i 进行更新。如果某个样本点已经被准确地分类，那么在训练下一个弱分类器时，就会降低它的权值；相反，如果该个样本点没有被准确地分类，就会提高它的权值。
3. 将各个弱分类器加权平均得到强分类器。误差率 e 低的弱分类器权重 α 较大，误差率 e 高的弱分类器权重 α 较小。

相信大家看完以上算法，会有以下三个疑问：

- 如何定义弱分类器的误差率 e？
- 如何定义弱学习器的权重 α？
- 如何定义样本的权重 D？

下面我们来回答以上三个问题。

假设我们的训练集样本是

$$T = \{(x, y_1), (x_2, y_2), \dots (x_m, y_m)\}$$

训练集的在第 k 个弱学习器的输入样本权重为：

$$D(k) = (w_{k1}, w_{k2}, \dots, w_{km})$$

其中基分类器的输入样本权重初始化为：

$$D(1) = (w_{11}, w_{12}, \dots, w_{1m}) = (\frac{1}{m}, \frac{1}{m}, \dots, \frac{1}{m})$$

1.2 Adaboost 三个问题解答

假设我们是二元分类问题，输出为 $\{-1, 1\}$，

- 对于第一个问题，第 k 个弱分类器 $G_k(x)$ 误差率为

$$e_k = P(G_k(x_i) \neq y_i) = \sum_{i=1}^{m} w_{ki} I(G_k(x_i) \neq y_i)$$

其中 $I(G_k(x_i) \neq y_i) = \begin{cases} 1 & G_k(x_i) \neq y_i \\ 0 & G_k(x_i) = y_i \end{cases}$

Q1：上式中 e_k 的物理含义是什么？

$G_k(x)$ 在加权训练数据集上的分类误差率等于被 $G_k(x)$ 误分类样本的加权和。

- 对于第二个问题，第 k 个弱分类器 $G_k(x)$ 的权重系数为

$$\alpha_k = \frac{1}{2} log \frac{1-e_k}{e_k}$$

α_k 表示弱分类器 $G_k(x)$ 在最终强分类器中的重要性。从上式可以看出，分类误差率 e_k 越大，弱分类器权重 α_k 越小。误差率越小则弱分类器权重越大。

Q2：上式中 α_k 的公式是否似曾相似？

我们知道 e_k 是误差率，那么 $1-e_k$ 为准确率，那么 $\frac{1-e_k}{e_k}$ 为相对准确率，显然相对准确率越大，分类器的权重系数就越大。

在逻辑回归中我们知道 p_i 为某事件发生的概率，$1-p_i$ 为某事件不发生的概率，$odds = \frac{p}{1-p}$ 为事件发生的相对概率，也称为几率。

为什么要用log呢？想想我们的逻辑回归 $log(odds) = w^T x + b$，加上 log 之后，函数变成了特征的线性函数。具体原因下面会讲解。

Q3：α_k 的取值范围？

正常情况下模型的准确率都会大于0.5（想一想为什么呢），即 $1-e_k >= 0.5$，那么 $\frac{1-e_k}{e_k}$ 就会大于1，那么 α_k 的取值范围为 $[0, +\infty]$

- 对于第三个问题，假设第 k 个弱分类器的输入样本集权重为 $D(k) = (w_{k1}, w_{k2}, \dots w_{km})$，则对应的第 $k+1$ 个弱分类器的样本权重为

$$w_{k+1,i} = \frac{w_{ki} e^{-\alpha_k y_i G_k(x_i)}}{Z_k}$$

化简上式：

$$w_{k+1,i} = \begin{cases} \frac{w_{ki} e^{\alpha_k}}{Z_k} & G_k(x_i) \neq y_i \\ \frac{w_{ki} e^{-\alpha_k}}{Z_k} & G_k(x_i) = y_i \end{cases}$$

式中 Z_k 是规范化因子：

$$Z_k = \sum_{i=1}^{m} w_{ki} e^{-\alpha_k y_i G_k(x_i)}$$

从以上公式可以看出，相对于正确分类的样本而言，误分类的样本权值被放大为 $e^{2\alpha} = \frac{e_m}{1-e_m}$(这里解释了为什么 α_k 要用log定义，而且系数为$\frac{1}{2}$)。

Q4：权重修正系数 $e^{-\alpha_k y_i G_k(x_i)}$ 为什么要写成指数的形式？

这个太简单了，大家想想 softmax 的公式：

$$S_i = \frac{e^{V_i}}{\sum_j e^{V_j}}$$

是不是高度雷同？再想想softmax的用途，一切迎刃而解。

好了，我们再来讨论一下指数里面为什么多了个 α_k，我们知道 α_k 表示弱分类器的权重，这里添加上 α_k 意味着性能越差的弱分类器其样本权重的变化幅度较小，而性能越强的弱分类器其样本权重的变化幅度越大。

最终分类Adaboost对弱分类器加权平均法得到强分类器为

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

那么弱学习器的权重 α 和样本的权重 D 是怎么来的呢？我们下面来解释其来历。

1.3 AdaBoost 样本权重和弱学习器权重推导

Adaboost 是若干个弱分类器加权和而得到最终的强分类器，因此是一个加法模型。

Adaboost 利用前一个弱学习器的结果来更新后一个弱学习器的训练集权重，因此学习算法为前向分步学习算法。

Adaboost 为损失函数为指数函数的分类问题。

第 $k-1$ 轮的强学习器为：

$$f_{k-1}(x)=\sum_{i=1}^{k-1}\alpha_i G_i(x)$$

而第 k 轮的强学习器为 ：

$$f_k(x)=\sum_{i=1}^{k}\alpha_i G_i(x)$$

上两式相减可以得到 ：

$$f_k(x)=f_{k-1}(x)+\alpha_k G_k(x)$$

可见强学习器的确是通过前向分步学习算法一步步而得到的。

Adaboost 损失函数为指数函数，即定义损失函数为：

$$\underbrace{arg\ min}_{\alpha,G}\sum_{i=1}^{m}e^{-y_i f_k(x)}$$

利用前向分步学习算法的关系可以得到损失函数为：

$$(\alpha_k,G_k(x))=\underbrace{arg\ min}_{\alpha,G}\sum_{i=1}^{m}e^{-y_i(f_{k-1}(x)+\alpha G_k(x))}$$

令 $w'_{ki}=e^{-y_i f_{k-1}(x)}$，它的值不依赖于 α,G，因此与最小化无关，仅仅依赖于 $f_{k-1}(x)$，随着每一轮迭代而改变。

将这个式子带入损失函数，损失函数转化为：

$$(\alpha_k,G_k(x))=\underbrace{arg\ min}_{\alpha,G}\sum_{i=1}^{m}w'_{ki}e^{-y_i\alpha G_k(x)}$$

上式中 $e^{-y_i\alpha G_k(x)} = \begin{cases} w'_{ki}e^{\alpha_k} & G_k(x_i) \neq y_i \\ w'_{ki}e^{-\alpha_k} & G_k(x_i) = y_i \end{cases}$

首先，我们求 $G_k(x)$ ，可以把 α 看成常数，损失函数可理解为求弱分类器$G_k(x)$ 使得其误分类样本的数量尽可能的少。等价于下面这个式子：

$$G_k(x) = \underbrace{arg\,min}_{G} \sum_{i=1}^{m} w'_{ki} I(y_i \neq G(x_i))$$

化简损失函数：

$$\begin{aligned} \sum_{i=1}^{m} w'_{ki} e^{-y_i\alpha G_k(x)} &= \sum_{y_i = G_k(x_i)}^{m} w'_{ki} e^{-\alpha} + \sum_{y_i \neq G_k(x_i)}^{m} w'_{ki} e^{\alpha} \\ &= (e^{\alpha} - e^{-\alpha}) \sum_{i=1}^{m} w'_{ki} I(y_i \neq G(x_i)) - e^{-\alpha} \sum_{i=1}^{m} w'_{ki} \end{aligned}$$

将 $G_k(x)$带入上式，并对 α 求导，使其等于 0，得到：

$$(e^{\alpha} + e^{-\alpha}) \sum_{i=1}^{m} w'_{ki} I(y_i \neq G(x_i)) + e^{-\alpha} \sum_{i=1}^{m} w'_{ki} = 0$$

$$(e^{\alpha} + e^{-\alpha}) \frac{\sum_{i=1}^{m} w'_{ki} I(y_i \neq G(x_i))}{\sum_{i=1}^{m} w'_{ki}} + e^{-\alpha} = 0$$

分类误差率 e_k ：

$$e_k = \frac{\sum_{i=1}^{m} w'_{ki} I(y_i \neq G(x_i))}{\sum_{i=1}^{m} w'_{ki}} = \sum_{i=1}^{m} w_{ki} I(y_i \neq G(x_i))$$

则：

$$(e^{\alpha} + e^{-\alpha}) e_k + e^{-\alpha} = 0$$

求得：

$$\alpha_k = \frac{1}{2} log \frac{1 - e_k}{e_k}$$

最后看样本权重的更新。利用 $f_k(x) = f_{k-1}(x) + \alpha_k G_k(x)$ 和 $w'_{ki} = e^{-y_i f_{k-1}(x)}$ ，即可得：

$$\begin{aligned} w'_{k+1,i} &= e^{-y_i f_k(x)} \\ &= e^{-y_i (f_{k-1}(x) + \alpha_k G_k(x))} \\ &= e^{-y_i f_{k-1}(x) - y_i \alpha_k G_k(x)} \\ &= e^{-y_i f_{k-1}(x)} e^{-y_i \alpha_k G_k(x)} \\ &= w'_{ki} e^{-y_i \alpha_k G_k(x)} \end{aligned}$$

1.4 AdaBoost二元分类问题算法流程

这里我们对AdaBoost二元分类问题算法流程做一个总结。

输入为样本集 $T = \{(x,y_1),(x_2,y_2),...(x_m,y_m)\}$，输出为 $\{-1,+1\}$，弱分类器算法, 弱分类器迭代次数 K。

输出为最终的强分类器 $f(x)$

1. 初始化样本集权重为

$$D_1 = (w_{11},w_{12},...w_{1m});\ \ w_{1i} = \frac{1}{m};\ \ i = 1,2...m$$

2. 对于 $k = 1,2,...K$：
   - 使用具有权重 D_k 的样本集来训练数据，得到弱分类器 $G_k(x)$
   - 计算 $G_k(x)$ 的分类误差率

$$e_k = P(G_k(x_i) \neq y_i) = \sum_{i=1}^{m} w_{ki} I(G_k(x_i) \neq y_i)$$

   - 计算弱分类器的系数

$$\alpha_k = \frac{1}{2} log \frac{1-e_k}{e_k}$$

   - 更新样本集的权重分布

$$w_{k+1,i} = \frac{w_{ki}}{Z_K} e^{-\alpha_k y_i G_k(x_i)}\ \ i = 1,2,...m$$

     这里 Z_k 是规范化因子

$$Z_k = \sum_{i=1}^{m} w_{ki} e^{-\alpha_k y_i G_k(x_i)}$$

3. 构建最终分类器为：

$$f(x) = sign(\sum_{k=1}^{K} \alpha_k G_k(x))$$

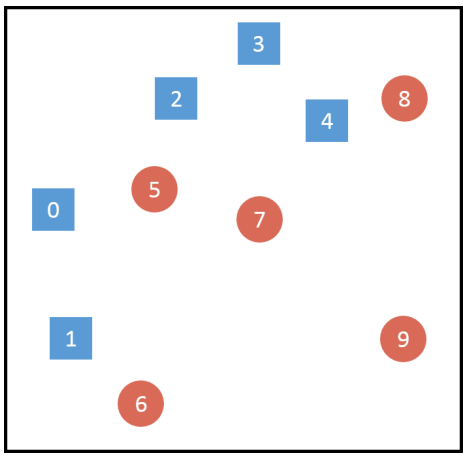
对于Adaboost多元分类算法，其实原理和二元分类类似，最主要区别在弱分类器的系数上。比如Adaboost SAMME算法，它的弱分类器的系数

$$\alpha_k = \frac{1}{2} log \frac{1-e_k}{e_k} + log(R-1)$$

其中R为类别数。从上式可以看出，如果是二元分类，R=2，则上式和我们的二元分类算法中的弱分类器的系数一致。

1.6 Adaboost算法案例

下面举一个简单的例子来看一下AdaBoost的实现过程：



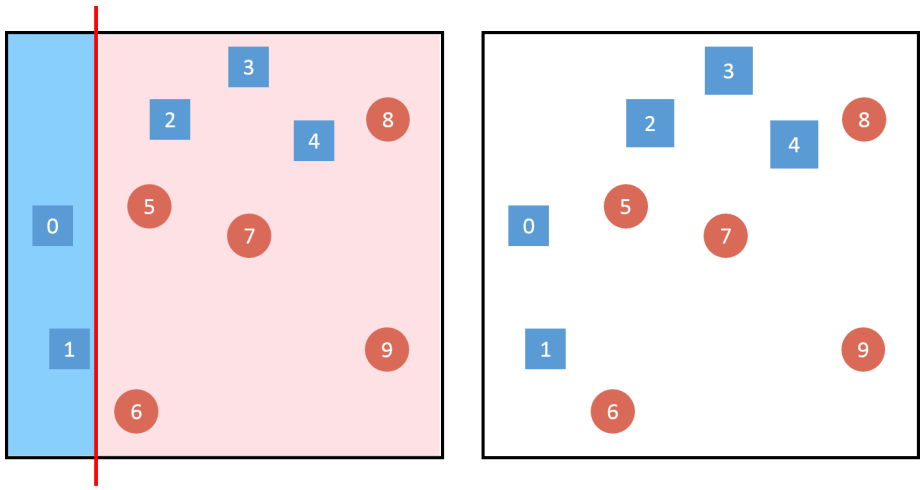
图中总共有10个样本，蓝色的正方形和红色的圆形分别表示两种类别，数字表示样本的序号 (本例中样本序号以0开始)。在这个过程中，使用水平或者垂直的直线作为分类器进行分类，以下为AdaBoost的步骤：

0. 初始化数据权重分布：

一共有10个样本，因此每个样本的权重均为$\frac{1}{10}$，即：

$$\begin{aligned} D_1 &= (w_{10}, w_{11}, w_{12}, w_{13}, w_{14}, w_{15}, w_{16}, w_{17}, w_{18}, w_{19}) \\ &= (0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1) \end{aligned}$$

1. 根据数据权重 D_1 得到一个基分类器 h_1 ：



上图为 h_1 的分类结果，其中样本 $2, 3, 4$被分错的，因此错误率 e_1 ：

$$e_1 = \sum_{i=0}^{9} w_{1i} I(G_1(x_i) \neq y_i) = w_{12} + w_{13} + w_{14} = 0.1 + 0.1 + 0.1 = 0.3$$

基分类器 h_1 的权重 α_1 ：

$$\alpha_1 = \frac{1}{2} log \frac{1 - e_1}{e_1} = \frac{1}{2} log \frac{1 - 0.3}{0.3} = 0.4236$$

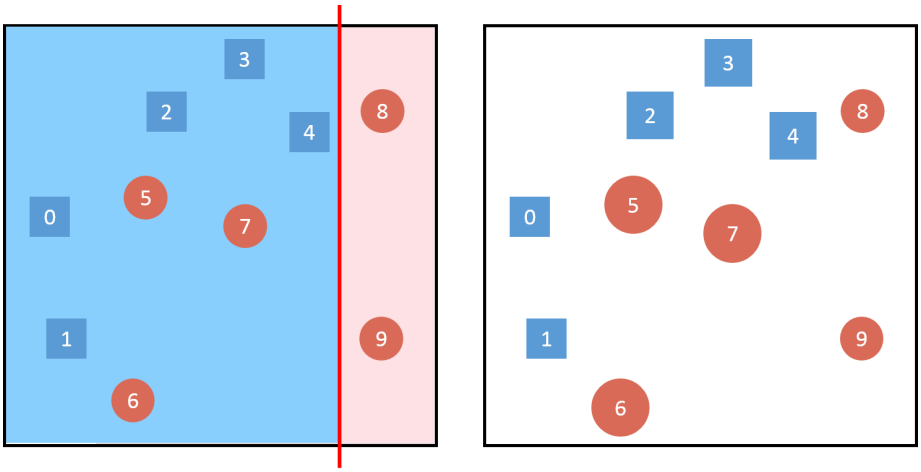
样本 $2, 3, 4$被分错的，应增加样本 $2, 3, 4$ 的权值（在右图中，表示样本 $2, 3, 4$ 面积变大表示对样本权值增大），得到一个新的样本权重分布 D_2 ：

$$\begin{aligned} D_2' &= (w_{10}e^{-\alpha_1}, w_{11}e^{-\alpha_1}, w_{12}e^{\alpha_1}, w_{13}e^{\alpha_1}, w_{14}e^{\alpha_1}, w_{15}e^{-\alpha_1}, w_{16}e^{-\alpha_1}, w_{17}e^{-\alpha_1}, w_{18}e^{-\alpha_1}, w_{19}e^{-\alpha_1}) \\ &= (0.6547, 0.6547, 1.5275, 1.5275, 1.5275, 0.6547, 0.6547, 0.6547, 0.6547, 0.6547) \end{aligned}$$

$$
\begin{aligned}
Z_2 &= w_{10}e^{-\alpha_1} + w_{11}e^{-\alpha_1} + w_{12}e^{\alpha_1} + w_{13}e^{\alpha_1} + w_{14}e^{\alpha_1} \\
&+ w_{15}e^{-\alpha_1} + w_{16}e^{-\alpha_1} + w_{17}e^{-\alpha_1} + w_{18}e^{-\alpha_1} + w_{19}e^{-\alpha_1} \\
&= 9.1652
\end{aligned}
$$

$$
\begin{aligned}
D_2 &= \frac{D'_2}{Z_2} \\
&= (w_{20}, w_{21}, w_{22}, w_{23}, w_{24}, w_{25}, w_{26}, w_{27}, w_{28}, w_{29}) \\
&= (0.07143, 0.07143, 0.1667, 0.1667, 0.1667, 0.07143, 0.07143, 0.07143, 0.07143, 0.07143)
\end{aligned}
$$

2. 根据 D_2 得到一个基分类器 h_2 ：



上图为h_2的分类结果，其中样本 $5, 6, 7$被分错的，因此错误率 e_2 ：

$$
e_2 = \sum_{i=0}^{9} w_{2i} I(G_2(x_i) \neq y_i) = w_{25} + w_{26} + w_{27} = 0.07143 + 0.07143 + 0.07143 = 0.2143
$$

基分类器 h_2 的权重 α_2 ：

$$
\alpha_2 = \frac{1}{2} log \frac{1 - e_2}{e_2} = \frac{1}{2} log \frac{1 - 0.2143}{0.2143} = 0.6496
$$

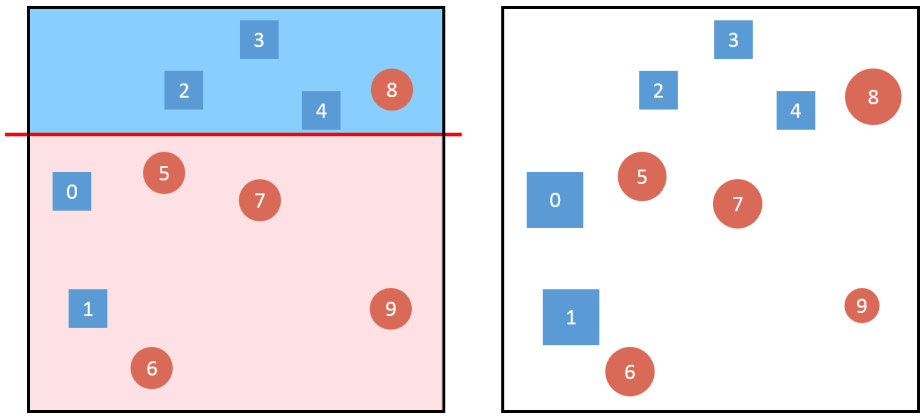
样本 $5, 6, 7$被分错的，应增加样本 $5, 6, 7$ 的权值（在右图中，表示样本 面积 $5, 6, 7$ 变大表示对样本权值增大），得到一个新的样本权重分布 D_3 ：

$$
\begin{aligned}
D'_3 &= (w_{20}e^{-\alpha_2}, w_{21}e^{-\alpha_2}, w_{22}e^{\alpha_2}, w_{23}e^{\alpha_2}, w_{24}e^{\alpha_2}, w_{25}e^{-\alpha_2}, w_{26}e^{-\alpha_2}, w_{27}e^{-\alpha_2}, w_{28}e^{-\alpha_2}, w_{29}e^{-\alpha_2}) \\
&= (0.0373, 0.0373, 0.0870, 0.0870, 0.0870, 0.1368, 0.1368, 0.1368, 0.0373, 0.0373)
\end{aligned}
$$

$$
\begin{aligned}
Z_3 &= w_{20}e^{-\alpha_2} + w_{21}e^{-\alpha_2} + w_{22}e^{\alpha_2} + w_{23}e^{\alpha_2} + w_{24}e^{\alpha_2} \\
&+ w_{25}e^{-\alpha_2} + w_{26}e^{-\alpha_2} + w_{27}e^{-\alpha_2} + w_{28}e^{-\alpha_2} + w_{29}e^{-\alpha_2} \\
&= 0.8207
\end{aligned}
$$

$$
\begin{aligned}
D_3 &= \frac{D'_3}{Z_3} \\
&= (w_{30}, w_{31}, w_{32}, w_{33}, w_{34}, w_{35}, w_{36}, w_{37}, w_{38}, w_{39}) \\
&= (0.0455, 0.0455, 0.1061, 0.1061, 0.1061, 0.1667, 0.1667, 0.1667, 0.0455, 0.0455)
\end{aligned}
$$

3. 根据 D_3 得到一个基分类器 h_3 ：



上图为h_3的分类结果，其中样本 $0,1,8$ 被分错的，因此图中错误率 e_3 ：

$$e_3 = \sum_{i=0}^{9} w_{3i} I(G_3(x_i) \neq y_i) = w_{30} + w_{31} + w_{38} = 0.0455 + 0.0455 + 0.0455 = 0.1364$$

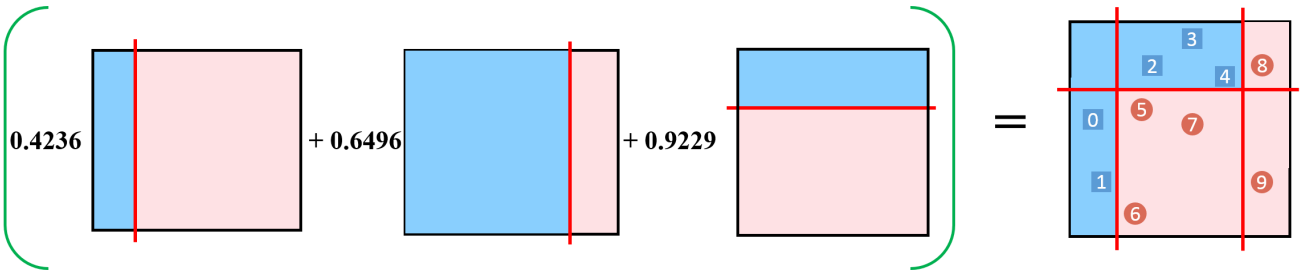
基分类器 h_3 的权重 α_3 ：

$$\alpha_3 = \frac{1}{2} log \frac{1 - e_3}{e_3} = \frac{1}{2} log \frac{1 - 0.1364}{0.1364} = 0.9229$$

样本 $0,1,8$ 被分错的，应增加样本 $0,1,8$ 的权值（在右图中，表示样本 面积 $0,1,8$ 变大表示对样本权值增大），得到一个新的样本权重分布 D_4 ：

$$\begin{aligned} D_4 &= (w_{40}, w_{41}, w_{42}, w_{43}, w_{44}, w_{45}, w_{46}, w_{47}, w_{48}, w_{49}) \\ &= (0.1667, 0.1667, 0.0614, 0.0614, 0.0614, 0.0965, 0.0965, 0.0965, 0.1667, 0.0263) \end{aligned}$$

4. 整合所有的子分类器：



1.7 Adaboost算法的参数解析

```
AdaBoostClassifier(base_estimator=None, n_estimators=50, learning_rate=1.0, algorithm='SAMME.R',
                   random_state=None)
```

框架参数 ：

1. base_estimator：弱学习器。

理论上可以选择任何一个分类或者回归学习器，不过需要支持样本权重。默认是决策树，另外需要注意，如果在algorithm中选择SAMME.R，则要求弱分类学习器还需要支持概率预测（predict_proba）。通常我们只要取默认值即可。

2. algorithm：这个参数只有AdaBoostClassifier有。

有两种取值：SAMME和SAMME.R。两者的主要区别是弱学习器权重的度量，SAMME用对样本集分类效果作为弱学习器权重，而SAMME.R使用了对样本集分类的预测概率大小来作为弱学习器权重。SAMME.R迭代一般比SAMME快，但是使用了SAMME.R， 则弱分类学习器参数base_estimator必须限制使用支持概率预测的分类器。一般使用默认的SAMME.R即可。

3. n_estimators： 弱学习器的最大迭代次数，或者说最大的弱学习器的个数。

n_estimators太小，容易欠拟合，n_estimators太大，容易过拟合，一般选择一个适中的数值。默认是50。

4. learning_rate: 即每个弱学习器的权重缩减系数 ν，默认是1

为了防止Adaboost过拟合，我们通常也会加入正则化项，定义权重缩减系数ν，修正弱学习器的迭代，则有

$$f_k(x) = f_{k-1}(x) + \nu\alpha_k G_k(x)$$

ν 的取值范围为 $0 < \nu \leq 1$。较小的 ν 意味着需要更多的弱学习器进行迭代。通常我们用步长和迭代最大次数一起来决定算法的拟合效果。

通常我们用步长和迭代最大次数一起来决定算法的拟合效果。所以n_estimators和learning_rate要一起调参。

假如我们采用默认的决策树，需要调参的还有决策树的一些超参数：max_depth, min_samples_split, min_samples_leaf, max_leaf_nodes等。